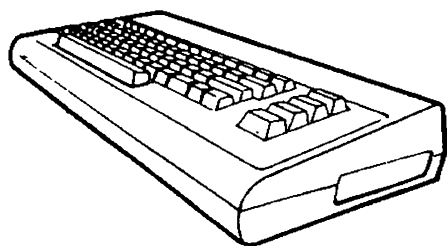


IL
S.O.
DEL
C=BM
64



E . U . M

IL SISTEMA OPERATIVO DEL COMMODORE 64E



EVM COMPUTERS

© Copyright 1983 E.V.M. Computers

Tutti i diritti sono riservati. Nessuna parte di questo libro può essere riprodotta posta in sistemi di archiviazione elettronici meccanici o fotocopiata senza autorizzazione scritta.

I EDIZIONE OTTOBRE 1983

E.V.M. Computers
Via Marconi 9/a
52025 MONTEVARCHI (AR)

INTRODUZIONE

Per la prima volta sui Personal Computer viene fornita l' elenco disassemblato del Sistema Operativo. Si tratta in effetti di un' elenco di routines e locazioni che permettono per la prima volta a chiunque desideri operare con questo computer di poter effettuare un lavoro serio e soprattutto di non essere costretti a perdere giorni e giorni di tentativi e prove.

Naturalmente e' richiesta una certa conoscenza del linguaggio macchina e dell' Assembler per il quale ci permettiamo di consigliare il volume CORSO DI ASSEMBLER PER IL CBM64 che viene consegnato insieme ad una cassetta (o disco a richiesta) contenente il Monitor, l' Assembler, il Disassembler ecc.

Lo scopo di questo libro e' quello di fornire una lista dettagliata del contenuto delle ROM e quindi di TUTTO il Sistema Operativo del computer CBM64.

Poiche' esistono due aree di ROM, che vanno rispettivamente da A000 a BFFF e da E000 a FFFF e che possono essere disabilitate separatamente, i listati che costituiscono la parte centrale del volume, sono stati distinti in due parti ad ognuna delle quali e' allegato anche il relativi Cross Reference.

A parte i listati relativi alle pagine 0/1/2/3, le colonne devono essere lette nella seguente maniera:

1. Indirizzo
2. Codice oggetto
3. Numero di linea
4. Label
5. Istruzione mnemonica
6. Operando
7. Commenti

Tutte le Label hanno un suffisso di quattro caratteri esadecimali (due per la pagina zero) che rappresentano l' indirizzo attuale.

Il prefisso di un un solo carattere puo' avere il seguente significato:

- B - Branch label
- J - Jump label
- S - Subroutine label
- T - Table (data) label

W - Word label
Z - Zero page label
X - External label

Alla fine di ognuno dei due listati, rispettivamente dell' Interprete Basic e del Sistema Operativo vero e proprio si trova un Cross Reference. Per ciascuna Label, messe in ordine alfabetico, si trova l' indirizzo della Label stessa e tutti i numeri di salto della Label.

A) Le pagine 1-3 contengono la parte del Sistema Operativo relativo alle pagine 0/1/2/3 della memoria utilizzata dall' Interprete Basic.

B) Da pagina 5 a 105 e' presente il listato disassemblato e commentato dell' Inteprete Basic, mentre da pagina 106 a 121 il relativo Cross Reference.

C) Da pagina 122 a pagina 126 sono presenti le pagine 0/1/2/3 della memoria utilizzate dal Sistema Operativo (parte KERNAL).

D) Da pagina 127 a pagina 222 le periferiche di I/O e le routines Kernal disassemblate. Da pagina 223 a 224 i Vettori Kernal.

E) Le pagine da 225 a 239 contengono il Cross Reference relativo alle routines Kernal mentre da pagina 239 a 242 il Cross Reference della pagina zero usata dal Sistema Operativo.

Inoltre in Appendice sono riportate sia le locazioni delle pagine 0/1/2/3/4 le routines mnemoniche relative al CBM64, al VIC ed alle serie 3000 e 4000.

Lo stesso e' stato fatto per routines del Sistema Operativo con un breve commento.

```

1      .L
2      .H
3 ;CBM-64-Part One
4 ;
5 ;
0000 6 Z00 = $00 ;6510 data direction register
0001 7 Z01 = $01 ;6510 I/O register
8 ; bit 0 (output) 0=RAM at $A000-$BFFF (BASIC area)
9 ; bit 1 (output) 0=RAM at $E000-$EFFF (Kernal area)
10 ; bit 2 (output) 0=access CRT shapes at $D000-$DFFF
11 ; bit 3 (output) cassette write line
12 ; bit 4 (input) cassette sense line
13 ; bit 5 (output) cassette motor control
14 ; bit 6 unused
15 ; bit 7 unused
0002 16 Z02 = $02 ;dummy address for offset
0003 17 Z03 = $03 ;fixed-float vector
0004 18 Z04 = $04 ;high byte of same
0007 19 Z07 = $07 ;separator/terminator/work field
0008 20 Z08 = $08 ;terminator/AND work field
0009 21 Z09 = $09 ;character position for TAB
000B 22 Z0B = $0B ;length BASIC line/AND-OR switch/# DIM
000C 23 Z0C = $0C ;reference/declaraction flag
000D 24 Z0D = $0D ;type: FF=string, 00=numeric
000E 25 Z0E = $0E ;type: 80=integer, 00=floating point
000F 26 Z0F = $0F ;DATA/string/error flag
0010 27 Z10 = $10 ;subscript/in flag/integers-arrays flag
0011 28 Z11 = $11 ;0=input, $40=get, $98=read
0012 29 Z12 = $12 ;(<=> operator
0013 30 Z13 = $13 ;CMD file number
0014 31 Z14 = $14 ;integer value (work)
0015 32 Z15 = $15 ;high byte of same
0016 33 Z16 = $16 ;string descriptor stack index
0017 34 Z17 = $17 ;previous string descriptor stack index
0018 35 Z18 = $18 ;high byte of same
0019 36 Z19 = $19 ;bottom of string descriptor stack
0022 37 Z22 = $22 ;utility pointer area
0023 38 Z23 = $23 ; "
0024 39 Z24 = $24 ; "
0025 40 Z25 = $25 ; "
0026 41 Z26 = $26 ;product for multiplication
0027 42 Z27 = $27 ; "
0028 43 Z28 = $28 ; "
0029 44 Z29 = $29 ; "
002B 45 Z2B = $2B ;pointer to start of BASIC
002C 46 Z2C = $2C ;high byte of same
002D 47 Z2D = $2D ;pointer start of variables
002E 48 Z2E = $2E ;high byte of same
002F 49 Z2F = $2F ;pointer to start of arrays
0030 50 Z30 = $30 ;high byte of same
0031 51 Z31 = $31 ;pointer to end of arrays
0032 52 Z32 = $32 ;high byte of same
0033 53 Z33 = $33 ;pointer to start of string storage
0034 54 Z34 = $34 ;high byte of same
0035 55 Z35 = $35 ;utility string pointer
0036 56 Z36 = $36 ;high byte of same
0037 57 Z37 = $37 ;pointer to limit of memory
0038 58 Z38 = $38 ;high byte of same
0039 59 Z39 = $39 ;current BASIC line number

```

003A	60	Z3A	=	\$3A	;high byte of same
003B	61	Z3B	=	\$3B	;previous BASIC line number
003C	62	Z3C	=	\$3C	;high byte of same
003D	63	Z3D	=	\$3D	;pointer to BASIC statement for CONT
003E	64	Z3E	=	\$3E	;high byte of same
003F	65	Z3F	=	\$3F	;current DATA line number
0040	66	Z40	=	\$40	;high byte of same
0041	67	Z41	=	\$41	;current DATA address
0042	68	Z42	=	\$42	;high byte of same
0043	69	Z43	=	\$43	;temporary read pointer
0044	70	Z44	=	\$44	;high byte of same
0045	71	Z45	=	\$45	;current variable name
0046	72	Z46	=	\$46	;second byte of same
0047	73	Z47	=	\$47	;current variable address
0048	74	Z48	=	\$48	;high byte of same
0049	75	Z49	=	\$49	;variable pointer for FOR/NEXT
004A	76	Z4A	=	\$4A	;high byte of same
004B	77	Z4B	=	\$4B	;save area
004C	78	Z4C	=	\$4C	;high byte of same
004D	79	Z4D	=	\$4D	;comparison symbol accumulator
004E	80	Z4E	=	\$4E	;misc. work area
004F	81	Z4F	=	\$4F	; " "
0050	82	Z50	=	\$50	; " "
0051	83	Z51	=	\$51	; " "
0053	84	Z53	=	\$53	; " "
0055	85	Z55	=	\$55	; " "
0056	86	Z56	=	\$56	; " "
0057	87	Z57	=	\$57	;misc. numeric work area
0058	88	Z58	=	\$58	; " "
0059	89	Z59	=	\$59	; " "
005A	90	Z5A	=	\$5A	; " "
005B	91	Z5B	=	\$5B	; " "
005C	92	Z5C	=	\$5C	; " "
005D	93	Z5D	=	\$5D	; " "
005E	94	Z5E	=	\$5E	; " "
005F	95	Z5F	=	\$5F	; " "
0060	96	Z60	=	\$60	; " "
0061	97	Z61	=	\$61	;floating point accu # 1 - exponent
0062	98	Z62	=	\$62	;flp # 1 - mantissa
0063	99	Z63	=	\$63	; " "
0064	100	Z64	=	\$64	; " "
0065	101	Z65	=	\$65	; " "
0066	102	Z66	=	\$66	;flp # 1 - sign
0067	103	Z67	=	\$67	;saved sign of flp accu
0068	104	Z68	=	\$68	;flp accu # 1 padding
0069	105	Z69	=	\$69	;flp accu # 2 thru Z6E
006A	106	Z6A	=	\$6A	
006B	107	Z6B	=	\$6B	
006C	108	Z6C	=	\$6C	
006D	109	Z6D	=	\$6D	
006E	110	Z6E	=	\$6E	
006F	111	Z6F	=	\$6F	;sign comparison accu #1 vs accu # 2
0070	112	Z70	=	\$70	;work pointer/guard bit
0071	113	Z71	=	\$71	;output index
0072	114	Z72	=	\$72	;high byte of same
007A	115	Z7A	=	\$7A	;current character address
007B	116	Z7B	=	\$7B	;high byte of same

0054	118 X0054 =	\$54	;JMP vector for functions
0073	119 X0073 =	\$73	;get next character
0079	120 X0079 =	\$79	;get current character
0080	121 X0080 =	\$80	;check for numeric character
0100	122 X0100 =	\$0100	;bottom of stack
0101	123 X0101 =	\$0101	;work area for fip to string conversion
0102	124 X0102 =	\$0102	; "
0103	125 X0103 =	\$0103	; "
0104	126 X0104 =	\$0104	; "
01FE	127 X01FE =	\$01FE	;line number for line in input buffer
01FF	128 X01FF =	\$01FF	;high byte of same
0200	129 X0200 =	\$0200	;BASIC input buffer
	130 ;		
	131 ;Operating System vector table		
	132 ;		
0300	133 X0300 =	\$0300	;error message link, std value = \$E38B
0302	134 X0302 =	\$0302	;BASIC warm start vector, std = \$A483
0304	135 X0304 =	\$0304	;crunch BASIC tokens, std = \$A57C
0306	136 X0306 =	\$0306	;print tokens vector, std value = \$A71A
0308	137 X0308 =	\$0308	;execute stat vector, std value = \$A7E4
030A	138 X030A =	\$030A	;get arithmetic element, std = \$AE86
0310	139 X0310 =	\$0310	;USR JMP vector, std value = \$B248
0314	140 W0314 =	\$0314	;IRQ vector, std value = \$EA31
0316	141 W0316 =	\$0316	;BRK vector, std value = \$FE66
0318	142 W0318 =	\$0318	;NMI vector, std value = \$FE47
031A	143 W031A =	\$031A	;OPEN vector, std value = \$F34A
031C	144 W031C =	\$031C	;CLOSE vector, std value = \$F291
031E	145 W031E =	\$031E	;set input vector, std value = \$F20E
0320	146 W0320 =	\$0320	;set output vector, std value = \$F250
0322	147 W0322 =	\$0322	;restore I/O vector, std value = \$F333
0324	148 W0324 =	\$0324	;INPUT vector, std value = \$F157
0326	149 W0326 =	\$0326	;OUTPUT vector, std value = \$F1CA
0328	150 W0328 =	\$0328	;test Stop Key vector, std value = \$F6ED
032A	151 W032A =	\$032A	;GET vector, std value = \$F13E
032C	152 W032C =	\$032C	;close files and channels, std = \$132F
032E	153 W032E =	\$032E	;unused vector, std value = \$FE66 (BRK)
0330	154 W0330 =	\$0330	;Load RAM vector, std value = \$F4A5
0332	155 W0332 =	\$0332	;Save RAM vector, std value = \$F5ED
	156 ;		
033C	157	.OR \$33C	
033C	158	.DS 192	;cassette buffer
03FC	159	.DS 4	
0400	160	.DS 1000	;video RAM
07E8	161	.DS 24	;sprite pointers
0800	162	.DS \$8800	;standard BASIC text area

9FEA	164 X9FEA =	\$9FEA	;address to access function jump table
E000	165 XE000 =	\$E000	;continuation of RND routine
E043	166 XE043 =	\$E043	;compute odd degrees for SIN and ATN
E097	167 WE097 =	\$E097	;RND command
E10C	168 XE10C =	\$E10C	;output a character
E112	169 XE112 =	\$E112	;input a character
E118	170 XE118 =	\$E118	;set output device
E11E	171 XE11E =	\$E11E	;set input device
E124	172 XE124 =	\$E124	;get a character from current device
E12A	173 WE12A =	\$E12A	;SYS command
E156	174 WE156 =	\$E156	;SAVE command
E165	175 WE165 =	\$E165	;VERIFY command
E168	176 WE168 =	\$E168	;LOAD command
E18E	177 WE18E =	\$E18E	;OPEN command
E1C7	178 WE1C7 =	\$E1C7	;CLOSE command
E264	179 WE264 =	\$E264	;COS command
E26B	180 WE26B =	\$E26B	;SIN command
E2B4	181 WE2B4 =	\$E2B4	;TAN command
E30E	182 WE30E =	\$E30E	;ATN command
E37B	183 WE37B =	\$E37B	;Warm Start entry
E386	184 XE386 =	\$E386	;print message READY
E394	185 WE394 =	\$E394	;RESET routine
FF90	186 XFF90 =	\$FF90	;control kernal messages
FFB7	187 XFFB7 =	\$FFB7	;read I/O status word
FFCC	188 XFFCC =	\$FFCC	;restore I/O devices to default
FFDB	189 XFFDB =	\$FFDB	;set real time clock
FFDE	190 XFFDE =	\$FFDE	;read real time clock
FFE1	191 XFFE1 =	\$FFE1	;check Stop key
FFE7	192 XFFE7 =	\$FFE7	;Close all channels and files
FFFO	193 XFFFO =	\$FFFO	;Read/Set XY cursor position

```

A000      195      .OR $A000
A000 94E3      196      .W WE394      ;RESET address
A002 7BE3      197      .W WE37B      ;Warm Start address
      198 ;
      199 ;program identifier (not tested)
      200 ;
A004 43424D    201      .BY "C","B","M","B","A","S","I","C
      202 ;
      203 ;address table for BASIC commands
      204 ;address - 1 used since routines reached via RTS
      205 ;
A00C 30A8      206 TA00C .W WA831-1      ;END
A00E 41A7      207      .W WA742-1      ;FOR
A010 1DAD      208      .W WAD1E-1      ;NEXT
A012 F7A8      209      .W WA8F8-1      ;DATA
A014 A4AB      210      .W WABA5-1      ;INPUT#
A016 BEAB      211      .W WABBF-1      ;INPUT
A018 80B0      212      .W WB081-1      ;DIM
A01A 05AC      213      .W WAC06-1      ;READ
A01C A4A9      214      .W WA9A5-1      ;LET
A01E 9FA8      215      .W WA8A0-1      ;GOTO
A020 70A8      216      .W WA871-1      ;RUN
A022 27A9      217      .W WA928-1      ;IF
A024 1CA8      218      .W WA81D-1      ;RESTORE
A026 82A8      219      .W WA883-1      ;GOSUB
A028 D1A8      220      .W WA8D2-1      ;RETURN
A02A 3AA9      221      .W WA93B-1      ;REM
A02C 2EA8      222      .W WA82F-1      ;STOP
A02E 4AA9      223      .W WA94B-1      ;ON
A030 2CB8      224      .W WB82D-1      ;WAIT
A032 67E1      225      .W WE168-1      ;LOAD
A034 55E1      226      .W WE156-1      ;SAVE
A036 64E1      227      .W WE165-1      ;VERIFY
A038 B2B3      228      .W WB3B3-1      ;DEF
A03A 23B8      229      .W WB824-1      ;POKE
A03C 7FAA      230      .W WAA80-1      ;PRINT#
A03E 9FAA      231      .W WAAA0-1      ;PRINT
A040 56A8      232      .W WA857-1      ;CONT
A042 9BA6      233      .W WA69C-1      ;LIST
A044 5DA6      234      .W WA65E-1      ;CLR
A046 85AA      235      .W WAA86-1      ;CMD
A048 29E1      236      .W WE12A-1      ;SYS
A04A BDE1      237      .W WE1BE-1      ;OPEN
A04C C6E1      238      .W WE1C7-1      ;CLOSE
A04E 7AAB      239      .W WAB7B-1      ;GET
A050 41A6      240      .W WA642-1      ;NEW

```

```

242 ;address table for BASIC functions
243 ;
A052 39BC 244 .W WBC39 ;SGN
A054 CCBC 245 .W WBCCC ;INT
A056 58BC 246 .W WBC58 ;ABS
A058 1003 247 .W X0310 ;USR
A05A 7DB3 248 .W WB37D ;FRE
A05C 9EB3 249 .W WB39E ;POS
A05E 71BF 250 .W WBF71 ;SQR
A060 97E0 251 .W WE097 ;RND
A062 EAB9 252 .W WB9EA ;LOG
A064 EDBF 253 .W WBFED ;EXP
A066 64E2 254 .W WE264 ;COS
A068 6BE2 255 .W WE26B ;SIN
A06A B4E2 256 .W WE2B4 ;TAN
A06C 0EE3 257 .W WE30E ;ATN
A06E 0DB8 258 .W WB80D ;PEEK
A070 7CB7 259 .W WB77C ;LEN
A072 65B4 260 .W WB465 ;STR$
A074 ADB7 261 .W WB7AD ;VAL
A076 8BB7 262 .W WB78B ;ASC
A078 ECB6 263 .W WB6EC ;CHR$
A07A 00B7 264 .W WB700 ;LEFT$
A07C 2CB7 265 .W WB72C ;RIGHT$
A07E 37B7 266 .W WB737 ;MID$
267 ;
268 ;table of priorities and addresses
269 ;for diadic operators
270 ;
A080 79 271 TA080 .BY $79
A081 69B8 272 .W WB86A-1 ;plus
273 ;
A083 79 274 .BY $79
A084 52B8 275 .W WB853-1 ;minus
276 ;
A086 7B 277 .BY $7B
A087 2ABA 278 .W WBA2B-1 ;times
279 ;
A089 7B 280 .BY $7B
A08A 11BB 281 .W WBB12-1 ;divided by
282 ;
A08C 7F 283 .BY $7F
A08D 7ABF 284 .W WBF7B-1 ;raise to power
285 ;
A08F 50 286 .BY $50
A090 E8AF 287 .W WAFE9-1 ;logical AND
288 ;
A092 46 289 .BY $46
A093 E5AF 290 .W WAFE6-1 ;logical OR
291 ;
A095 7D 292 .BY $7D
A096 B3BF 293 .W WBFB4-1 ;monadic minus
294 ;
A098 5A 295 .BY $5A
A099 D3AE 296 .W WAED4-1 ;monadic NOT
297 ;
A09B 64 298 .BY $64
A09C 15B0 299 .W WB016-1 ;> = <

```

```

301 ;table of keywords
302 ;same sequence as address table IA00C
303 ;
AO9E 454EC4 304 TA09E .BY 'E','N','D'+$80
AOA1 464FD2 305 .BY 'F','O','R'+$80
AOA4 4E4558 306 .BY 'N','E','X','T'+$80
AOA8 444154 307 .BY 'D','A','T','A'+$80
AOAC 494E50 308 .BY 'I','N','P','U','T','#'+$80
AOB2 494E50 309 .BY 'I','N','P','U','T'+$80
AOB7 4449CD 310 .BY 'D','I','M'+$80
AOBA 524541 311 .BY 'R','E','A','D'+$80
AOBE 4C45D4 312 .BY 'L','E','T'+$80
AOC1 474F54 313 .BY 'G','O','T','O'+$80
AOC5 5255CE 314 .BY 'R','U','N'+$80
AOC8 49C6 315 .BY 'I','F'+$80
AOCA 524553 316 .BY 'R','E','S','T','O','R','E'+$80
AOD1 474F53 317 .BY 'G','O','S','U','B'+$80
AOD6 524554 318 .BY 'R','E','T','U','R','N'+$80
AODC 5245CD 319 .BY 'R','E','M'+$80
AODF 53544F 320 .BY 'S','T','O','P'+$80
AOE3 4FCE 321 .BY 'O','N'+$80
AOE5 574149 322 .BY 'W','A','I','T'+$80
AOE9 4C4F41 323 .BY 'L','O','A','D'+$80
AOED 534156 324 .BY 'S','A','V','E'+$80
AOF1 564552 325 .BY 'V','E','R','I','F','Y'+$80
AOF7 4445C6 326 .BY 'D','E','F'+$80
AOFA 504F4B 327 .BY 'P','O','K','E'+$80
AOFE 505249 328 .BY 'P','R','I','N','T','#'+$80
A104 505249 329 .BY 'P','R','I','N','T'+$80
A109 434F4E 330 .BY 'C','O','N','T'+$80
A10D 4C4953 331 .BY 'L','I','S','T'+$80
A111 434CD2 332 .BY 'C','L','R'+$80
A114 434DC4 333 .BY 'C','M','D'+$80
A117 5359D3 334 .BY 'S','Y','S'+$80
A11A 4F5045 335 .BY 'O','P','E','N'+$80
A11E 434C4F 336 .BY 'C','L','O','S','E'+$80
A123 4745D4 337 .BY 'G','E','T'+$80
A126 4E45D7 338 .BY 'N','E','W'+$80

```



```

340 ;table of functions
341 ;
A129 544142 342 .BY ^T,^A,^B,^(+$80
A12D 54CF 343 .BY ^T,^O+$80
A12F 46CE 344 .BY ^F,^N+$80
A131 535043 345 .BY ^S,^P,^C,^(+$80
A135 544845 346 .BY ^T,^H,^E,^N+$80
A139 4E4FD4 347 .BY ^N,^O,^T+$80
A13C 535445 348 .BY ^S,^T,^E,^P+$80
A140 AB 349 .BY ^++$80
A141 AD 350 .BY ^+$80
A142 AA 351 .BY ^*+$80
A143 AF 352 .BY ^/+$80
A144 DE 353 .BY $DE ;exponent sign
A145 414EC4 354 .BY ^A,^N,^D+$80
A148 4FD2 355 .BY ^O,^R+$80
A14A BE 356 .BY ^>+$80
A14B BD 357 .BY ^-+$80
A14C BC 358 .BY ^<+$80
A14D 5347CE 359 .BY ^S,^G,^N+$80
A150 494ED4 360 .BY ^I,^N,^T+$80
A153 4142D3 361 .BY ^A,^B,^S+$80
A156 5553D2 362 .BY ^U,^S,^R+$80
A159 4652C5 363 .BY ^F,^R,^E+$80
A15C 504FD3 364 .BY ^F,^O,^S+$80
A15F 5351D2 365 .BY ^S,^Q,^R+$80
A162 524EC4 366 .BY ^R,^N,^D+$80
A165 4C4FC7 367 .BY ^L,^O,^C+$80
A168 4558D0 368 .BY ^E,^X,^P+$80
A16B 434FD3 369 .BY ^C,^O,^S+$80
A16E 5349CE 370 .BY ^S,^I,^N+$80
A171 5441CE 371 .BY ^T,^A,^N+$80
A174 4154CE 372 .BY ^A,^T,^N+$80
A177 504545 373 .BY ^P,^E,^E,^K+$80
A17B 4C45CE 374 .BY ^L,^E,^N+$80
A17E 535452 375 .BY ^S,^T,^R,^$+$80
A182 5641CC 376 .BY ^V,^A,^L+$80
A185 4153C3 377 .BY ^A,^S,^C+$80
A188 434852 378 .BY ^C,^H,^R,^$+$80
A18C 4C4546 379 .BY ^L,^E,^F,^T,^$+$80
A191 524947 380 .BY ^R,^I,^G,^H,^T,^$+$80
A197 4D4944 381 .BY ^M,^I,^D,^$+$80
382 ;
383 ;other commands
384 ;
A19B 47CF 385 .BY ^G,^O+$80
A19D 00 386 .BY $00 ;end of keywords

```

```

388 ;error messages
389 ;
A19E 544F4F 390 TA19E .BY T,O,O,M,A,N,Y,F,I,L,E,S+$80
A1AC 46494C 391 TA1AC .BY F,I,L,E,O,P,E,N+$80
A1B5 46494C 392 TA1B5 .BY F,I,L,E,N,O,T,O,P,E,N+$80
A1C2 46494C 393 TA1C2 .BY F,I,L,E,N,O,T,F,O,U,N,D+$80
A1D0 444556 394 TA1D0 .BY D,E,V,I,C,E,N,O,T,
A1DB 505245 395 .BY P,R,E,S,E,N,T+$80
A1E2 4E4F54 396 TA1E2 .BY N,O,T,I,N,P,U,T,F,I,L,E+$80
A1F0 4E4F54 397 TA1F0 .BY N,O,T,O,U,T,P,U,T,F,I,L,E+$80
A1FF 4D4953 398 TA1FF .BY M,I,S,S,I,N,G,F,I,L,E,
A20C 4E414D 399 .BY N,A,M,E+$80
A210 494C4C 400 TA210 .BY I,L,L,E,G,A,L,D,E,V,I,C,E,
A21F 4E554D 401 .BY N,U,M,B,E,R+$80
A225 4E4558 402 TA225 .BY N,E,X,T,W,I,T,H,O,U,T
A231 20464F 403 .BY F,O,R+$80
A235 53594E 404 TA235 .BY S,Y,N,T,A,X+$80
A23B 524554 405 TA23B .BY R,E,T,U,R,N,W,I,T,H,O,U,T,
A24A 474F53 406 .BY G,O,S,U,B+$80
A24F 4F5554 407 TA24F .BY O,U,T,O,F,D,A,T,A+$80
A25A 494C4C 408 TA25A .BY I,L,L,E,G,A,L
A261 205155 409 .BY Q,U,A,N,T,I,T,Y+$80
A26A 4F5645 410 TA26A .BY O,V,E,R,F,L,O,W+$80
A272 4F5554 411 TA272 .BY O,U,T,O,F,M,E,M,O,R,Y+$80
A27F 554E44 412 TA27F .BY U,N,D,E,F,$27,D,
A287 535441 413 .BY S,T,A,T,E,M,E,N,T+$80
A290 424144 414 TA290 .BY B,A,D,S,U,B,S,C,R,I,P,T+$80
A29D 524544 415 TA29D .BY R,E,D,I,M,$27,D,A,R,R,A,Y+$80
A2AA 444956 416 TA2AA .BY D,I,V,I,S,I,O,N,B,Y
A2B5 205A45 417 .BY Z,E,R,O+$80
A2BA 494C4C 418 TA2BA .BY I,L,L,E,G,A,L,D,I,R,E,C,T+$80
A2C8 545950 419 TA2C8 .BY T,Y,P,E,M,I,S,M,A,T,C,H+$80
A2D5 535452 420 TA2D5 .BY S,T,R,I,N,G,T,O,O,L,O,N,G+$80
A2E4 46494C 421 TA2E4 .BY F,I,L,E,D,A,T,A+$80
A2ED 464F52 422 TA2ED .BY F,O,R,M,U,L,A,I,O,O,
A2F9 434F4D 423 .BY C,O,M,P,L,E,X+$80
A300 43414E 424 TA300 .BY C,A,N,$27,T,C,O,N,T,I,N,U,E+$80
A30E 554E44 425 TA30E .BY U,N,D,E,F,$27,D
A315 204655 426 .BY F,U,N,C,T,I,O,N+$80
A31E 564552 427 TA31E .BY V,E,R,I,F,Y+$80
A324 4C4F41 428 TA324 .BY L,O,A,D+$80

```

```

430 ;error message address table
431 ;index 01-1E loaded into X to reference message table
432 ;
A328 9EA1 433 TA328 .W TA19E ;01 Too Many Files
A32A ACA1 434 .W TA1AC ;02 File Open
A32C B5A1 435 .W TA1B5 ;03 File Not Open
A32E C2A1 436 .W TA1C2 ;04 File Not Found
A330 D0A1 437 .W TA1D0 ;05 Device Not Present
A332 E2A1 438 .W TA1E2 ;06 Not Input File
A334 FOA1 439 .W TA1F0 ;07 Not Output File
A336 FFA1 440 .W TA1FF ;08 Missing File Name
A338 10A2 441 .W TA210 ;09 Illegal Device Number
A33A 25A2 442 .W TA225 ;0A Next Without For
A33C 35A2 443 .W TA235 ;0B Syntax
A33E 3BA2 444 .W TA23B ;0C Return Without Gosub
A340 4FA2 445 .W TA24F ;0D Out Of Data
A342 5AA2 446 .W TA25A ;0E Illegal Quantity
A344 6AA2 447 .W TA26A ;0F Overflow
A346 72A2 448 .W TA272 ;10 Out Of Memory
A348 7FA2 449 .W TA27F ;11 Undef'd Statement
A34A 90A2 450 .W TA290 ;12 Bad Subscript
A34C 9DA2 451 .W TA29D ;13 Redim'd Array
A34E AAA2 452 .W TA2AA ;14 Division By Zero
A350 BAA2 453 .W TA2BA ;15 Illegal Direct
A352 C8A2 454 .W TA2C8 ;16 Type Mismatch
A354 D5A2 455 .W TA2D5 ;17 String Too Long
A356 E4A2 456 .W TA2E4 ;18 File Data
A358 EDA2 457 .W TA2ED ;19 Formula Too Complex
A35A 00A3 458 .W TA300 ;1A Can't Continue
A35C 0EA3 459 .W TA30E ;1B Undef'd Function
A35E 1EA3 460 .W TA31E ;1C Verify
A360 24A3 461 .W TA324 ;1D Load
A362 83A3 462 .W TA383 ;1E Break
463 ;
A364 0D4F4B 464 TA364 .BY $OD,`O`,`K,$OD,$00
A369 202045 465 TA369 .BY ``,``,`E`,`R`,`R`,`O`,`R,$00
A371 20494E 466 TA371 .BY ``,``,`I`,`N`,``,`,$00
A376 0D0A52 467 TA376 .BY $OD,$0A,`R`,`E`,`A`,`D`,`Y`,``,`,$OD,$0A,$00
A381 0D0A 468 TA381 .BY $OD,$0A
A383 425245 469 TA383 .BY `B`,`R`,`E`,`A`,`K,$00,$A0

```

```

471 ;search for "FOR" blocks on stack (code 81)
472 ;
473 ;if Z49/Z4A > $00FF, get FOR block with address = Z49/Z4A
474 ;if Z49/Z4A < $0100, get highest FOR block in Z49/Z4A
475 ;upon exit, X = stack above FOR block
476 ;Z=1 when FOR block found
477 ;Z=0 when FOR block not found
478 ;A=code of block found
479 ;
A38A BA 480 SA38A TSX
A38B E8 481 INX
A38C E8 482 INX
A38D E8 483 INX
A38E E8 484 INX ;point to highest block
A38F BD0101 485 BA38F LDA X0101,X
A392 C981 486 CMP $81 ;if no FOR block
A394 D021 487 BNE BA3B7 ;return with Z=0
A396 A54A 488 LDA Z4A
A398 D00A 489 BNE BA3A4 ;if no address
A39A BD0201 490 LDA X0102,X ;get address from highest block
A39D 8549 491 STA Z49
A39F BD0301 492 LDA X0103,X
A3A2 854A 493 STA Z4A
A3A4 DD0301 494 BA3A4 CMP X0103,X ;compare address to block
A3A7 D007 495 BNE BA3B0
A3A9 A549 496 LDA Z49
A3AB DD0201 497 CMP X0102,X
A3AE F007 498 BEQ BA3B7 ;return with Z=1 if equal
A3B0 8A 499 BA3B0 TXA ;if not equal
A3B1 18 500 CLC
A3B2 6912 501 ADC $12 ;skip a FOR block
A3B4 AA 502 TAX
A3B5 D0D8 503 BNE BA38F ;and try again
A3B7 60 504 BA3B7 RTS

```

```

506 ;move bytes after a check for space
507 ;
A3B8 2008A4 508 SA3B8 JSR SA408 ;do array area overflow check for AY
A3BB 8531 509 STA Z31 ;set new end of array area
A3BD 8432 510 STY Z32
511 ;
512 ;move bytes routine
513 ;source address in Z5F/Z60
514 ;end of source address in Z5A/Z5B
515 ;end of destination address in Z58/Z59
516 ;
517 ;highest bytes moved first
518 ;X and Y at 0 upon return
519 ;
A3BF 38 520 SA3BF SEC
A3C0 A55A 521 LDA Z5A
A3C2 E55F 522 SBC Z5F
A3C4 8522 523 STA Z22 ;compute length of area to be moved
A3C6 A8 524 TAY ;Y=low order 8 bits
A3C7 A55B 525 LDA Z5B
A3C9 E560 526 SBC Z60
A3CB AA 527 TAX ;X=high order 8 bits
A3CC E8 528 INX
A3CD 98 529 TYA
A3CE F023 530 BEQ BA3F3 ;if multiple of 256, skip remainder
A3D0 A55A 531 LDA Z5A
A3D2 38 532 SEC
A3D3 E522 533 SBC Z22
A3D5 855A 534 STA Z5A
A3D7 B003 535 BCS BA3DC ;Z5A/Z5B points to start source block
A3D9 C65B 536 DEC Z5B
A3DB 38 537 SEC
A3DC A558 538 BA3DC LDA Z58
A3DE E522 539 SBC Z22
A3E0 8558 540 STA Z58
A3E2 B008 541 BCS BA3EC
A3E4 C659 542 DEC Z59 ;Z58/Z59 points to destination
A3E6 9004 543 BCC BA3EC
A3E8 B15A 544 BA3E8 LDA (Z5A),Y ;fetch a byte from source
A3EA 9158 545 STA (Z58),Y ;move it to destination
A3EC 88 546 BA3EC DEY
A3ED D0F9 547 BNE BA3E8 ;repeat for all 256 bytes in a page
A3EF B15A 548 LDA (Z5A),Y ;also last byte
A3F1 9158 549 STA (Z58),Y
A3F3 C65B 550 BA3F3 DEC Z5B ;point to lower block of source
A3F5 C659 551 DEC Z59 ;point to lower block of destination
A3F7 CA 552 DEX
A3F8 D0F2 553 BNE BA3EC ;repeat for all blocks
A3FA 60 554 RTS

```

```

556 ;test for 2 * A bytes available on the stack
557 ;
558 ;allow 62 bytes extra for interrupt processing
559 ;indicate fatal error if not enough space left
560 ;
A3FB 0A 561 SA3FB ASL A ;multiply A * 2
A3FC 693E 562 ADC $3E ;62 bytes extra
A3FE B035 563 BCS BA435 ;error if overflow
A400 8522 564 STA Z22
A402 BA 565 TSX
A403 E422 566 CPX Z22 ;if stack above result
A405 902E 567 BCC BA435 ;overflow error
A407 60 568 RTS
569 ;
570 ;array area overflow check
571 ;
A408 C434 572 SA408 CPY Z34 ;if A below string storage pointer
A40A 9028 573 BCC BA434 ;return
A40C D004 574 BNE BA412
A40E C533 575 CMP Z33
A410 9022 576 BCC BA434
A412 48 577 BA412 PHA ;save AY
A413 A209 578 LDX $09
A415 98 579 TYA
A416 48 580 BA416 PHA
A417 B557 581 LDA Z57,X ;save Z58-Z61 on stack
A419 CA 582 DEX
A41A 10FA 583 BPL BA416
A41C 2026B5 584 JSR SB526 ;perform string cleanup
A41F A2F7 585 LDX $F7
A421 68 586 BA421 PLA
A422 9561 587 STA Z61,X ;restore Z58-Z61 from stack
A424 E8 588 INX
A425 30FA 589 BMI BA421
A427 68 590 PLA
A428 A8 591 TAY
A429 68 592 PLA ;restore AY
A42A C434 593 CPY Z34 ;if AY still above string pointer,
A42C 9006 594 BCC BA434 ;then error OUT OF MEMORY
A42E D005 595 BNE BA435
A430 C533 596 CMP Z33
A432 B001 597 BCS BA435
A434 60 598 BA434 RTS
599 ;
600 ;fatal error OUT OF MEMORY
601 ;
A435 A210 602 BA435 LDX $10 ;point to error OUT OF MEMORY

```

```

        604 ;handle error messages
        605 ;
        606 ;X points to error message address table TA328
        607 ;
A437 6C0003 608 JA437 JMP (X0300) ;handle error message (normally A43A)
        609 ;
        610 ;standard error message handler
        611 ;
A43A 8A 612 TXA
A43B 0A 613 ASL A
A43C AA 614 TAX
A43D BD26A3 615 LDA TA328-2,X ;fetch address from error message table
A440 8522 616 STA Z22
A442 BD27A3 617 LDA TA328-1,X
A445 8523 618 STA Z23
A447 20CCFF 619 JSR XFFCC ;restore I/O devices to default
A44A A900 620 LDA $00
A44C 8513 621 STA Z13 ;set default CMD output file
A44E 20D7AA 622 JSR SAAD7 ;end line on CMD output file
A451 2045AB 623 JSR SAB45 ;print question mark on output file
A454 A000 624 LDY $00
A456 B122 625 BA456 LDA (Z22),Y
A458 48 626 PHA
A459 297F 627 AND $7F
A45B 2047AB 628 JSR SAB47 ;print a character on output file
A45E C8 629 INY
A45F 68 630 PLA
A460 10F4 631 BPL BA456 ;repeat if end of message not reached
A462 207AA6 632 JSR SA67A ;reset stack and program pointers
A465 A969 633 LDA <TA369
A467 A0A3 634 LDY >TA369 ;set AY to message ERROR
A469 201EAB 635 JA469 JSR SAB1E ;print message from AY
A46C A43A 636 LDY Z3A ;if in direct mode
A46E C8 637 INY
A46F F003 638 BEQ BA474 ;restart BASIC
A471 20C2BD 639 JSR SBDC2 ;else print IN STATEMENT and #
A474 A976 640 BA474 LDA <TA376
A476 A0A3 641 LDY >TA376 ;set AY to message READY
A478 201EAB 642 JSR SAB1E ;print READY message
A47B A980 643 LDA $80
A47D 2090FF 644 JSR XFF90 ;set kernal msg flag for direct mode
A480 6C0203 645 BA480 JMP (X0302) ;perform warm start (normally A483)
        646 ;
        647 ;standard warm start routine (BASIC Interpreter)
        648 ;
A483 2060A5 649 JSR SA560 ;get statement into the input buffer
A486 867A 650 STX Z7A ;move beginning of line address
A488 847B 651 STY Z7B ;to current character address
A48A 207300 652 JSR X0073 ;get next character of statement
A48D AA 653 TAX
A48E F0F0 654 BEQ BA480 ;skip statement if end of line
A490 A2FF 655 LDX $FF
A492 863A 656 STX Z3A ;set Run/Direct switch to Direct
A494 9006 657 BCC BA49C ;if no line number,
A496 2079A5 658 JSR SA579 ;encode keywords in statement
A499 4CE1A7 659 JMP JA7E1 ;and go execute statement

```

```

661 ;numbered statements
662 ;
A49C 206BA9 663 BA49C JSR SA96B ;gather line number into Z14/Z15
A49F 2079A5 664 JSR SA579 ;encode keywords in statement
A4A2 840B 665 STY Z0B ;save index for end of line
A4A4 2013A6 666 JSR SA613 ;search for statement in program
A4A7 9044 667 BCC BA4ED ;if not found, insert line
668 ;
669 ;delete old statement (indexed from Z5F/Z60)
670 ;
A4A9 A001 671 LDY $01
A4AB B15F 672 LDA (Z5F),Y ;move low of pointer to next statement
A4AD 8523 673 STA Z23
A4AF A52D 674 LDA Z2D ;move low of ptr beyond last statement
A4B1 8522 675 STA Z22
A4B3 A560 676 LDA Z60 ;move byte of ptr to current statement
A4B5 8525 677 STA Z25
A4B7 A55F 678 LDA Z5F ;low byte of ptr to current statement
A4B9 88 679 DEY ;low byte of pointer to next statement
A4BA F15F 680 SBC (Z5F),Y ;length of gap in 2's complement
A4BC 18 681 CLC
A4BD 652D 682 ADC Z2D ;+ low byte ptr beyond last statement
A4BF 852D 683 STA Z2D ;= new pointer beyond last statement
A4C1 8524 684 STA Z24 ;also saved
A4C3 A52E 685 LDA Z2E ;high byte of ptr beyond last statement
A4C5 69FF 686 ADC $FF ;+ carry
A4C7 852E 687 STA Z2E ;= new high of pointer beyond last stmt
A4C9 E560 688 SBC Z60 ;high of pointer to current stmt
A4CB AA 689 TAX ;high byte of # of bytes to move
A4CC 38 690 SEC
A4CD A55F 691 LDA Z5F ;low byte of ptr to current statement
A4CF E52D 692 SBC Z2D ;low of pointer beyond last statement
A4D1 A8 693 TAY ;number of bytes to move
A4D2 B003 694 BCS BA4D7 ;if necessary,
A4D4 E8 695 INX ;adjust carry
A4D5 C625 696 DEC Z25 ;also adjust low end pointer
A4D7 18 697 BA4D7 CLC
A4D8 6522 698 ADC Z22
A4DA 9003 699 BCC BA4DF
A4DC C623 700 DEC Z23
A4DE 18 701 CLC
A4DF B122 702 BA4DF LDA (Z22),Y ;get byte from low end
A4E1 9124 703 STA (Z24),Y ;move to new low end
A4E3 C8 704 INY
A4E4 D0F9 705 BNE BA4DF ;repeat until page moved
A4E6 E623 706 INC Z23 ;update page pointers
A4E8 E625 707 INC Z25
A4EA CA 708 DEX
A4EB D0F2 709 BNE BA4DF ;repeat until last block moved

```


	711	;insert new line at location indicated by Z5F/Z60	
	712	;	
A4ED 2059A6	713	BA4ED JSR SA659	;reset program pointers
A4F0 2033A5	714	JSR SA533	;relink BASIC lines
A4F3 AD0002	715	LDA X0200	;if first byte of new line is a null,
A4F6 F088	716	BEQ BA480	;don't insert
A4F8 18	717	CLC	
A4F9 A52D	718	LDA Z2D	;get low byte of pointer to end of line
A4FB 855A	719	STA Z5A	;save as end pointer for move
A4FD 650B	720	ADC Z0B	;add length of line to be inserted
A4FF 8558	721	STA Z58	;save as end output pointer for move
A501 A42E	722	LDY Z2E	;save high bytes also
A503 845B	723	STY Z5B	
A505 9001	724	BCC BA508	
A507 C8	725	INY	
A508 8459	726	BA508 STY Z59	
A50A 20B8A3	727	JSR SA3B8	;move bytes and check for overflow
A50D A514	728	LDA Z14	;move line number
A50F A415	729	LDY Z15	
A511 8DFE01	730	STA X01FE	;to fields preceeding input buffer
A514 8CFF01	731	STY X01FF	
A517 A531	732	LDA Z31	;use new upper limit
A519 A432	733	LDY Z32	
A51B 852D	734	STA Z2D	;as new end of program
A51D 842E	735	STY Z2E	
A51F A40B	736	LDY Z0B	;get length of line to be inserted
A521 88	737	DEY	
A522 B9FC01	738	BA522 LDA X0200-4,Y	;move new line
A525 915F	739	STA (Z5F),Y	;into gap
A527 88	740	DEY	
A528 10F8	741	BPL BA522	;loop until line moved
A52A 2059A6	742	JSR SA659	;reset program pointers
A52D 2033A5	743	JSR SA533	;relink BASIC lines
A530 4C80A4	744	JMP BA480	;go back to BASIC

```

746 ;relink BASIC
747 ;
A533 A52B 748 SA533 LDA Z2B ;get pointer to beginning of BASIC text
A535 A42C 749 LDY Z2C
A537 8522 750 STA Z22 ;hold in temporary pointer
A539 8423 751 STY Z23
A53B 18 752 CLC
A53C A001 753 BA53C LDY $01
A53E B122 754 LDA (Z22),Y ;get byte at current statement + 1
A540 F01D 755 BEQ BA55F ;if at 0, end of program reached
A542 A004 756 LDY $04 ;set start index
A544 C8 757 BA544 INY ;increment index
A545 B122 758 LDA (Z22),Y
A547 D0FB 759 BNE BA544 ;scan to end of line
A549 C8 760 INY
A54A 98 761 TYA
A54B 6522 762 ADC Z22 ;compute pointer to next statement
A54D AA 763 TAX
A54E A000 764 LDY $00
A550 9122 765 STA (Z22),Y ;set link in beginning of statement
A552 A523 766 LDA Z23
A554 6900 767 ADC $00
A556 C8 768 INY
A557 9122 769 STA (Z22),Y ;set new current statement address
A559 8622 770 STX Z22
A55B 8523 771 STA Z23
A55D 90DD 772 BCC BA53C ;repeat until all statements processed
A55F 60 773 BA55F RTS
774 ;
775 ;get statement into input buffer
776 ;
A560 A200 777 SA560 LDX $00 ;start with index=0
A562 2012E1 778 BA562 JSR XE112 ;get clean character from CMD file
A565 C90D 779 CMP $0D ;if Return,
A567 F00D 780 BEQ BA576 ;go end line
A569 9D0002 781 STA X0200,X ;else store character in input buffer
A56C E8 782 INX
A56D E059 783 CPX $59 ;if statement not too long
A56F 90F1 784 BCC BA562 ;repeat
A571 A217 785 LDX $17 ;else point to message STRING TOO LONG
A573 4C37A4 786 JMP JA437 ;and print error message
787 ;
A576 4CCAAA 788 BA576 JMP JAACA ;go end line

```

```

A579 6C0403 790 SA579 JMP (X0304) ;crunch tokens (normally A57C)
791 ;
792 ;crunch tokens routine
793 ;
A57C A67A 794 LDX Z7A ;get input index
A57E A004 795 LDY $04 ;set output index -5
A580 840F 796 STY ZOF ;set not a DATA statement
A582 BD0002 797 BA582 LDA X0200,X ;get next character
A585 1007 798 BPL BA58E
A587 C9FF 799 CMP $FF ;check for PI
A589 F03E 800 BEQ BA5C9
A58B E8 801 INX ;other codes above $7F are ignored
A58C D0F4 802 BNE BA582
A58E C920 803 BA58E CMP $20 ;move blanks
A590 F037 804 BEQ BA5C9
A592 8508 805 STA Z08 ;save possible quote
A594 C922 806 CMP ""
A596 F056 807 BEQ BA5EE ;move string when between quotes
A598 240F 808 BIT ZOF ;move character if in DATA statement
A59A 702D 809 EVS BA5C9
A59C C93F 810 CMP $3F ;if question mark
A59E D004 811 BNE BA5A4
A5A0 A999 812 LDA $99 ;replace by code for PRINT
A5A2 D025 813 BNE BA5C9
A5A4 C930 814 BA5A4 CMP "0 ;move if numeric
A5A6 9004 815 BCC BA5AC
A5A8 C93C 816 CMP $3C
A5AA 901D 817 BCC BA5C9
A5AC 8471 818 BA5AC STY Z71 ;save output index
A5AE A000 819 LDY $00 ;initialize table index
A5B0 840B 820 STY Z0B ;initialize keyword count
A5B2 88 821 DEY
A5B3 867A 822 STX Z7A ;save input index
A5B5 CA 823 DEX
A5B6 C8 824 BA5B6 INY ;next character in table
A5B7 E8 825 INX ;next character in line
A5B8 BD0002 826 BA5B8 LDA X0200,X ;get character from input buffer
A5BB 38 827 SEC
A5BC F99EAD 828 SBC TA09E,Y ;if equal to table character
A5BF F0F5 829 BEQ BA5B6 ;try next
A5C1 C980 830 CMP $80 ;if equal to character without bit 7
A5C3 D030 831 BNE BA5F5
A5C5 050B 832 ORA Z0B ;save keyword count + $80
A5C7 A471 833 BA5C7 LDY Z71 ;restore output index
A5C9 E8 834 BA5C9 INX ;advance input index
A5CA C8 835 INY ;advance output index
A5CB 99FB01 836 STA X0200-5,Y ;store character or code
A5CE B9FB01 837 LDA X0200-5,Y
A5D1 F036 838 BEQ BA609 ;if zero, end of string
A5D3 38 839 SEC
A5D4 E93A 840 SBC " ;if colon, set flag to not data
A5D6 F004 841 BEQ BA5DC
A5D8 C949 842 CMP $49 ;if code for DATA, set flag
A5DA D002 843 BNE BA5DE
A5DC 850F 844 BA5DC STA ZOF
A5DE 38 845 BA5DE SEC
A5DF E955 846 SBC $55 ;if REM
A5E1 D09F 847 BNE BA582
A5E3 8508 848 STA Z08 ;set 00 as terminator
A5E5 BD0002 849 BA5E5 LDA X0200,X ;get next character from input buffer

```

A5E8 F0DF	850	BEQ BA5C9	;end upon zero
A5EA C508	851	CMP Z08	
A5EC F0DB	852	BEQ BA5C9	;or terminator (")
A5EE C8	853 BA5EE	INY	;advance output index
A5EF 99FB01	854	STA X0200-5,Y	;move character
A5F2 E8	855	INX	;advance input index
A5F3 D0F0	856	BNE BA5E5	;repeat
A5F5 A67A	857 BA5F5	LDX Z7A	;at end of keyword from table
A5F7 E60B	858	INC Z0B	;increment count
A5F9 C8	859 BA5F9	INY	;advance table pointer
A5FA B99DA0	860	LDA TA09E-1,Y	;beyond end of last keyword
A5FD 10FA	861	BPL BA5F9	
A5FF B99EA0	862	LDA TA09E,Y	;if not at end of table
A602 D0B4	863	BNE BA5B8	
A604 BD0002	864	LDA X0200,X	;restore input character
A607 10BE	865	BPL BA5C7	;repeat
A609 99FD01	866 BA609	STA X0200-3,Y	;set end of line
A60C C67B	867	DEC Z7B	;set current character pointer
A60E A9FF	868	LDA \$FP	
A610 857A	869	STA Z7A	;to beginning of input buffer again
A612 60	870	RTS	

```

872 ;search for a statement in the program
873 ;
874 ;line number search argument in Z14/Z15
875 ;exit with C=1/0 for found/not found
876 ;Z5F/Z60 set to address of statement found, or next higher
877 ;
A613 A52B 878 SA613 LDA Z2B ;get pointer to beginning of BASIC
A615 A62C 879 LDX Z2C
A617 A001 880 BA617 LDY $01
A619 855F 881 STA Z5F ;save pointer to current statement
A61B 8660 882 STX Z60
A61D B15F 883 LDA (Z5F),Y ;get link address in current statement
A61F F01F 884 BEQ BA640 ;0=end of program
A621 C8 885 INY
A622 C8 886 INY
A623 A515 887 LDA Z15
A625 D15F 888 CMP (Z5F),Y ;compare statement numbers
A627 9018 889 BCC BA641 ;return if search high (not found)
A629 F003 890 BEQ BA62E ;since high matches, check low
A62B 88 891 DEY
A62C D009 892 BNE BA637 ;continue since statement number low
A62E A514 893 BA62E LDA Z14
A630 88 894 DEY
A631 D15F 895 CMP (Z5F),Y ;compare low order of statement numbers
A633 900C 896 BCC BA641 ;return if high
A635 F00A 897 BEQ BA641 ;return if equal
A637 88 898 BA637 DEY
A638 B15F 899 LDA (Z5F),Y ;get pointer to next statement in AY
A63A AA 900 TAX
A63B 88 901 DEY
A63C B15F 902 LDA (Z5F),Y
A63E B0D7 903 BCS BA617 ;repeat
A640 18 904 BA640 CLC
A641 60 905 BA641 RTS

```

```

907 ;"NEW" command
908 ;
A642 D0FD 909 WA642 BNE BA641 ;if parameters present, SYNTAX Error
A644 A900 910 LDA $00
A646 A8 911 TAY
A647 912B 912 STA (Z2B),Y ;store zero's
A649 C8 913 INY
A64A 912B 914 STA (Z2B),Y ;in first 2 bytes of BASIC program area
A64C A52B 915 LDA Z2B ;pointer to start of BASIC
A64E 18 916 CLC
A64F 6902 917 ADC $02 ;+ 2
A651 852D 918 STA Z2D ;becomes pointer to end of BASIC
A653 A52C 919 LDA Z2C
A655 6900 920 ADC $00
A657 852E 921 STA Z2E
A659 208EA6 922 SA659 JSR SA68E ;re-initialize current character ptr
A65C A900 923 LDA $00 ;Z = 1
924 ;
925 ;"CLR" command
926 ;
A65E D02D 927 WA65E BNE BA68D ;if parameters present, SYNTAX Error
A660 20E7FF 928 SA660 JSR XFFE7 ;close all channels and files
A663 A537 929 LDA Z37 ;get memory limit in AY
A665 A438 930 LDY Z38
A667 8533 931 STA Z33 ;reset string storage pointer
A669 8434 932 STY Z34
A66B A52D 933 LDA Z2D ;move pointer to variables in AY
A66D A42E 934 LDY Z2E
A66F 852F 935 STA Z2F ;into pointer to start of arrays
A671 8430 936 STY Z30
A673 8531 937 STA Z31 ;also into pointer to end of arrays
A675 8432 938 STY Z32
A677 201DA8 939 JSR SA81D ;perform RESTORE
940 ;
941 ;reset stack and program pointers
942 ;
A67A A219 943 SA67A LDX <Z19
A67C 8616 944 STX Z16 ;reset string descriptor stack index
A67E 68 945 PLA ;get own return address in AY
A67F A8 946 TAY
A680 68 947 PLA
A681 A2FA 948 LDX $FA
A683 9A 949 TXS ;reset stack pointer
A684 48 950 PHA ;put own return address back on stack
A685 98 951 TYA
A686 48 952 PHA
A687 A900 953 LDA $00
A689 853E 954 STA Z3E ;clear pointer for CONTINUE
A68B 8510 955 STA Z10 ;also subscript flag
A68D 60 956 BA68D RTS

```

```

958 ;set current character pointer to beginning of program -1
959 ;
A68E 18 960 SA68E CLC
A68F A52B 961 LDA Z2B ;get pointer to beginning of BASIC area
A691 69FF 962 ADC $FF ;-1
A693 857A 963 STA Z7A ;store into current character pointer
A695 A52C 964 LDA Z2C ;same for high byte
A697 69FF 965 ADC $FF
A699 857B 966 STA Z7B
A69B 60 967 RTS
968 ;
969 ;"LIST" command
970 ;
A69C 9006 971 WA69C BCC BA6A4 ;parameter must be numeric,
A69E F004 972 BEQ BA6A4 ;or else none at all
A6A0 C9AB 973 CMP $AB ;or "-"
A6A2 DOE9 974 BNE BA68D ;if none of above, SYNTAX Error
A6A4 206BA9 975 BA6A4 JSR SA96B ;gather decimal number into Z14/Z15
A6A7 2013A6 976 JSR SA613 ;search for statement in program
A6AA 207900 977 JSR X0079 ;get current character
A6AD F00C 978 BEQ BA6BB ;if not at end of statement
A6AF C9AB 979 CMP $AB ;or "-"
A6B1 D08E 980 BNE BA641 ;then SYNTAX Error
A6B3 207300 981 JSR X0073 ;get next character after "-"
A6B6 206BA9 982 JSR SA96B ;gather decimal number into Z14/Z15
A6B9 D086 983 BNE BA641 ;if not end of statement, SYNTAX Error
A6BB 68 984 BA6BB PLA ;remove return address from stack
A6BC 68 985 PLA
A6BD A514 986 LDA Z14 ;if last parameter zero (missing)
A6BF 0515 987 ORA Z15
A6C1 D006 988 BNE BA6C9
A6C3 A9FF 989 LDA $FF
A6C5 8514 990 STA Z14 ;set it to 65535
A6C7 8515 991 STA Z15

```

```

993 ;list statement from Z5F/Z60 thru Z14/Z15
994 ;
A6C9 A001 995 BA6C9 LDY $01 ;set string switch to not a string
A6CB 840F 996 STY ZOF
A6CD B15F 997 LDA (Z5F),Y ;get high byte of link address for line
A6CF F043 998 BEQ BA714 ;0=end of program, go restart BASIC
A6D1 202CA8 999 JSR SA82C ;check for Stop Key
A6D4 20D7AA 1000 JSR SAAD7 ;end line on CMD output file
A6D7 C8 1001 INY
A6D8 B15F 1002 LDA (Z5F),Y
A6DA AA 1003 TAX ;set AX to statement #
A6DB C8 1004 INY
A6DC B15F 1005 LDA (Z5F),Y
A6DE C515 1006 CMP Z15 ;compare to end of list statement
A6E0 D004 1007 BNE BA6E6
A6E2 E414 1008 CPX Z14
A6E4 F002 1009 BEQ BA6E8
A6E6 B02C 1010 BA6E6 BCS BA714 ;if higher, stop
A6E8 8449 1011 BA6E8 STY Z49 ;save index in statement
A6EA 20CDBD 1012 JSR SBD CD ;print statement # from AX
A6ED A920 1013 LDA $20 ;insert a blank
A6EF A449 1014 BA6EF LDY Z49 ;restore index in statement
A6F1 297F 1015 AND $7F ;clear bit 7
A6F3 2047AB 1016 BA6F3 JSR SAB47 ;print character from A on CMD file
A6F6 C922 1017 CMP "" ;if a quote character
A6F8 D006 1018 BNE BA700
A6FA A50F 1019 LDA ZOF
A6FC 49FF 1020 EOR $FF ;flip string switch
A6FE 850F 1021 STA ZOF
A700 C8 1022 BA700 INY ;advance index
A701 F011 1023 BEQ BA714 ;if line too long, restart BASIC
A703 B15F 1024 LDA (Z5F),Y
A705 D010 1025 BNE BA717 ;if not end of line, go print it
A707 A8 1026 TAY ;if end of line,
A708 B15F 1027 LDA (Z5F),Y ;get pointer to next statement in AX
A70A AA 1028 TAX
A70B C8 1029 INY
A70C B15F 1030 LDA (Z5F),Y
A70E 865F 1031 STX Z5F ;and reset line address
A710 8560 1032 STA Z60
A712 D0B5 1033 BNE BA6C9 ;then loop back for next line
A714 4C86E3 1034 BA714 JMP XE386 ;go restart BASIC

```



```

A717 6C0603 1036 BA717 JMP (X0306) ;go print tokens (normally A71A)
              1037 ;
              1038 ;print tokens
              1039 ;
A71A 10D7 1040 BPL BA6F3 ;if bit 7 clear, print one character
A71C C9FF 1041 CMP $FF ;if code for PI,
A71E F0D3 1042 BEQ BA6F3 ;print PI
A720 240F 1043 BIT ZOF ;if in string mode
A722 30CF 1044 BMI BA6F3 ;also print shifted codes
A724 38 1045 SEC
A725 E97F 1046 SBC $7F ;compute keyword #
A727 AA 1047 TAX
A728 8449 1048 STY Z49 ;save index in statement
A72A A0FF 1049 LDY $FF
A72C CA 1050 BA72C DEX ;if keyword found
A72D F008 1051 BEQ BA737 ;go print keyword
A72F C8 1052 BA72F INY
A730 B99EA0 1053 LDA TA09E,Y ;else get character from keyword table
A733 10FA 1054 BPL BA72F
A735 30F5 1055 BMI BA72C ;until bit 7 set (end of keyword)
              1056 ;
              1057 ;print keyword
              1058 ;
A737 C8 1059 BA737 INY
A738 B99EA0 1060 LDA TA09E,Y ;get next character from keyword table
A73B 30B2 1061 BMI BA6EF ;if bit 7 set go print last character
A73D 2047AB 1062 JSR SAB47 ;print character on CMD file
A740 D0F5 1063 BNE BA737 ;and repeat

```

		1065 ;"FOR" command	
		1066 ;	
A742 A980	1067	WA742 LDA \$80	;set flag for
A744 8510	1068	STA Z10	;no integer variables or array elements
A746 20A5A9	1069	JSR SA9A5	;execute comand LET
A749 208AA3	1070	JSR SA38A	;get FOR block of variable used
A74C D005	1071	BNE BA753	
A74E 8A	1072	TXA	;if found,
A74F 690F	1073	ADC \$0F	;drop stack ptr below that FOR block
A751 AA	1074	TAX	
A752 9A	1075	TXS	
A753 68	1076	BA753 PLA	;remove return address from stack
A754 68	1077	PLA	
A755 A909	1078	LDA \$09	;9 addresses on stack needed
A757 20FBA3	1079	JSR SA3FB	;else OUT OF MEMORY Error
A75A 2006A9	1080	JSR SA906	;get offset to end of statement
A75D 18	1081	CLC	
A75E 98	1082	TYA	
A75F 657A	1083	ADC Z7A	;compute pointer to end of statement
A761 48	1084	PHA	
A762 A57B	1085	LDA Z7B	
A764 6900	1086	ADC \$00	
A766 48	1087	PHA	;save on stack
A767 A53A	1088	LDA Z3A	
A769 48	1089	PHA	
A76A A539	1090	LDA Z39	
A76C 48	1091	PHA	;save current statement # on stack
A76D A9A4	1092	LDA \$A4	
A76F 20FFAE	1093	JSR SAEFF	;if not T0, then SYNTAX Error
A772 208DAD	1094	JSR SAD8D	;initial value must be non-string
A775 208AAD	1095	JSR SAD8A	;get next non-string value (T0 value)
A778 A566	1096	LDA Z66	;move sign of flp accu
A77A 097F	1097	ORA \$7F	
A77C 2562	1098	AND Z62	
A77E 8562	1099	STA Z62	;into most significant bit
A780 A98B	1100	LDA <WA78B	
A782 A0A7	1101	LDY >WA78B	
A784 8522	1102	STA Z22	;set return address for continuation
A786 8423	1103	STY Z23	
A788 4C43AE	1104	JMP JAE43	;round and save flp accu (T0 value)
	1105 ;		
A78B A9BC	1106	WA78B LDA <TB9BC	;set AY to default step (1)
A78D A0B9	1107	LDY >TB9BC	
A78F 20A2BB	1108	JSR SBBA2	;load default step
A792 207900	1109	JSR X0079	;get next character
A795 C9A9	1110	CMP \$A9	;if STEP
A797 D006	1111	BNE BA79F	
A799 207300	1112	JSR X0073	;get next character
A79C 208AAD	1113	JSR SAD8A	;get next non-string value
A79F 202BBC	1114	BA79F JSR SBC2B	;get SGN of flp accu
A7A2 2038AE	1115	JSR SAE38	;round flp accu and save on stack
A7A5 A54A	1116	LDA Z4A	;save pointer to FOR variable
A7A7 48	1117	PHA	
A7A8 A549	1118	LDA Z49	
A7AA 48	1119	PHA	
A7AB A981	1120	LDA \$81	;save code for FOR block
A7AD 48	1121	PHA	

```

1123 ;execute next statement
1124 ;
A7AE 202CA8 1125 JA7AE JSR SA82C ;check Stop Key
A7B1 A57A 1126 LDA Z7A
A7B3 A47B 1127 LDY Z7B ;get ptr to end of old statement in AY
A7B5 C002 1128 CPY $02
A7B7 EA 1129 NOP
A7B8 F004 1130 BEQ BA7BE ;if not direct mode,
A7BA 853D 1131 STA Z3D ;store pointer for possible CONT
A7BC 843E 1132 STY Z3E
A7BE A000 1133 BA7BE LDY $00
A7C0 B17A 1134 LDA (Z7A),Y ;get last byte
A7C2 D043 1135 BNE BA807 ;if not 0, check for ":"
A7C4 A002 1136 LDY $02
A7C6 B17A 1137 LDA (Z7A),Y ;get high of pointer to next statement
A7C8 18 1138 CLC
A7C9 D003 1139 BNE BA7CE
A7CB 4C4BA8 1140 JMP JA84B ;0=end of program
1141 ;
A7CE C8 1142 BA7CE INY
A7CF B17A 1143 LDA (Z7A),Y ;move statement # into Z39/Z3A
A7D1 8539 1144 STA Z39
A7D3 C8 1145 INY
A7D4 B17A 1146 LDA (Z7A),Y
A7D6 853A 1147 STA Z3A
A7D8 98 1148 TTA ;advance pointer over statement header
A7D9 657A 1149 ADC Z7A
A7DB 857A 1150 STA Z7A
A7DD 9002 1151 BCC JA7E1
A7DF E67B 1152 INC Z7B
A7E1 6C0803 1153 JA7E1 JMP (X0308) ;go execute a statement (normally A7E4)
1154 ;
1155 ;execute a statement
1156 ;
A7E4 207300 1157 JSR X0073 ;get next character
A7E7 20EDA7 1158 JSR SA7ED ;execute command in A
A7EA 4CAEA7 1159 JMP JA7AE ;go execute next statement

```

```

1161 ;execute command in A
1162 ;
A7ED F03C 1163 SA7ED BEQ BA82B ;if Z = 1, return (dummy statement)
A7EF E980 1164 JA7EF SBC $80 ;if A < $80
A7F1 9011 1165 BCC BA804 ;perform LET
A7F3 C923 1166 CMP $23 ;if A = > A3 (highest command)
A7F5 B017 1167 BCS BA80E ;then check for GOTO
A7F7 0A 1168 ASL A ;get 2 * (A - 80) as index to table
A7F8 A8 1169 TAY
A7F9 B90DA0 1170 LDA TA00C+1,Y ;move routine address onto stack
A7FC 48 1171 PHA
A7FD B90CA0 1172 LDA TA00C,Y
A800 48 1173 PHA
A801 4C7300 1174 JMP X0073 ;get first char of parameters and go
1175 ;
A804 4CA5A9 1176 BA804 JMP SA9A5 ;execute command LET
1177 ;
A807 C93A 1178 BA807 CMP ': ;if character is a colon,
A809 F0D6 1179 BEQ JA7E1 ;execute next command
A80B 4C08AF 1180 BA80B JMP JAF08 ;else print SYNTAX Error
1181 ;
A80E C94B 1182 BA80E CMP $4B ;if not code for GO
A810 D0F9 1183 BNE BA80B ;then SYNTAX Error
A812 207300 1184 JSR X0073 ;else get next character
A815 A9A4 1185 LDA $A4 ;set code for TO
A817 20FFAE 1186 JSR SAEFF ;if equal
A81A 4CA0A8 1187 JMP JA8A0 ;go do command GOTO

```

```

1189 ;"RESTORE" command
1190 ;
A81D 1191 WA81D = *
A81D 38 1192 SA81D SEC
A81E A52B 1193 LDA Z2B ;get pointer to beginning of BASIC
A820 E901 1194 SBC $01 ;-1
A822 A42C 1195 LDY Z2C ;into AY
A824 B001 1196 BCS BA827
A826 88 1197 DEY
A827 8541 1198 BA827 STA Z41 ;store AY in current DATA address
A829 8442 1199 STY Z42
A82B 60 1200 BA82B RTS
1201 ;
A82C 20E1FF 1202 SA82C JSR XFFE1 ;test Stop Key
1203 ;
1204 ;"STOP" command
1205 ;
A82F B001 1206 WA82F BCS BA832 ;preserve C flag
1207 ;
1208 ;"END" command
1209 ;
A831 18 1210 WA831 CLC ;clear C flag
A832 D03C 1211 BA832 BNE BA870 ;if parameters present, SYNTAX Error
A834 A57A 1212 LDA Z7A ;set AY to address of current line
A836 A47B 1213 LDY Z7B
A838 A63A 1214 LDX Z3A
A83A E8 1215 INX
A83B F00C 1216 BEQ BA849 ;if not in direct mode,
A83D 853D 1217 STA Z3D ;save pointer for possible CONT
A83F 843E 1218 STY Z3E
A841 A539 1219 LDA Z39 ;AY=current statement #
A843 A43A 1220 LDY Z3A
A845 853B 1221 STA Z3B ;move current statement #
A847 843C 1222 STY Z3C ;to previous statement #
A849 68 1223 BA849 PLA ;remove own return address
A84A 68 1224 PLA
A84B A981 1225 JA84B LDA <TA381 ;set AY to BREAK message
A84D A0A3 1226 LDY >TA381
A84F 9003 1227 BCC BA854 ;if STOP command,
A851 4C69A4 1228 JMP JA469 ;print message and restart BASIC
A854 4C86E3 1229 BA854 JMP XE386 ;else restart BASIC

```

	1231	;"CONT" command	
	1232	;	
A857 D017	1233	WA857 BNE BA870	;if parameters present, SYNTAX Error
A859 A21A	1234	LDX \$1A	;point to message CAN'T CONTINUE
A85B A43E	1235	LDY Z3E	;get pointer to current statement in AY
A85D D003	1236	BNE BA862	;if pointer=0,
A85F 4C37A4	1237	JMP JA437	;fatal error
	1238	;	
A862 A53D	1239	BA862 LDA Z3D	;move saved character pointer
A864 857A	1240	STA Z7A	;into current character pointer
A866 847B	1241	STY Z7B	
A868 A53B	1242	LDA Z3B	;move previous statement #
A86A A43C	1243	LDY Z3C	
A86C 8539	1244	STA Z39	;into current statement #
A86E 843A	1245	STY Z3A	
A870 60	1246	BA870 RTS	
	1247	;	
	1248	;"RUN" command	
	1249	;	
A871 08	1250	WA871 PHP	
A872 A900	1251	LDA \$00	
A874 2090FF	1252	JSR XFF90	;set kernal messages flag to RUN mode
A877 28	1253	PLP	
A878 D003	1254	BNE BA87D	;if no parameters,
A87A 4C59A6	1255	JMP SA659	;do CLR and start program
	1256	;	
A87D 2060A6	1257	BA87D JSR SA660	;else do CLR and then
A880 4C97A8	1258	JMP JA897	;do GOTO
	1259	;	
	1260	;"GOSUB" command	
	1261	;	
A883 A903	1262	WA883 LDA \$03	;need 3 address on stack
A885 20FBA3	1263	JSR SA3FB	;else error OUT OF MEMORY
A888 A57B	1264	LDA Z7B	
A88A 48	1265	PHA	
A88B A57A	1266	LDA Z7A	;save pointer to current character
A88D 48	1267	PHA	;on the stack
A88E A53A	1268	LDA Z3A	
A890 48	1269	PHA	
A891 A539	1270	LDA Z39	
A893 48	1271	PHA	;save current statement # on stack
A894 A98D	1272	LDA \$8D	
A896 48	1273	PHA	;save code for GOSUB
A897 207900	1274	JA897 JSR X0079	;get current character
A89A 20A0A8	1275	JSR JA8A0	;execute command GOTO
A89D 4CAEA7	1276	JMP JA7AE	;execute next statement

	1278	;"GOTO" command	
	1279	;	
A8A0	1280	WA8A0 = *	
A8A0 206BA9	1281	JSR SA96B	;gather decimal number into Z14/Z15
A8A3 2009A9	1282	JSR SA909	;get offset to end of line in Y
A8A6 38	1283	SEC	
A8A7 A539	1284	LDA Z39	;fetch current statement number
A8A9 E514	1285	SBC Z14	
A8AB A53A	1286	LDA Z3A	
A8AD E515	1287	SBC Z15	;if forward jump,
A8AF B00B	1288	BCS BA8BC	;let current character pointer
A8B1 98	1289	TYA	;point to next statement
A8B2 38	1290	SEC	
A8B3 657A	1291	ADC Z7A	
A8B5 A67B	1292	LDX Z7B	;also start search there
A8B7 9007	1293	BCC BA8C0	
A8B9 E8	1294	INX	
A8BA B004	1295	BCS BA8C0	
A8BC A52B	1296	BA8BC LDA Z2B	;else start at beginning of BASIC
A8BE A62C	1297	LDX Z2C	
A8C0 2017A6	1298	BA8C0 JSR BA617	;search for statement starting at AX
A8C3 901E	1299	BCC BA8E3	;if not found, UNDEF'D STATEMENT
A8C5 A55F	1300	LDA Z5F	;move pointer to statement found
A8C7 E901	1301	SBC \$01	; -1
A8C9 857A	1302	STA Z7A	;into current character pointer
A8CB A560	1303	LDA Z60	
A8CD E900	1304	SBC \$00	
A8CF 857B	1305	STA Z7B	
A8D1 60	1306	BA8D1 RTS	
	1307	;	
	1308	;"RETURN" command	
	1309	;	
A8D2 D0FD	1310	WA8D2 BNE BA8D1	;if parameters present, SYNTAX Error
A8D4 A9FF	1311	LDA \$FF	
A8D6 854A	1312	STA Z4A	
A8D8 208AA3	1313	JSR SA38A	;get FOR block
A8DB 9A	1314	TXS	;remove everything above current block
A8DC C98D	1315	CMP \$8D	;and the block found must be a GOSUB
A8DE F00B	1316	BEQ BA8EB	
A8E0 A20C	1317	LDX \$0C	;else point to RETURN WITHOUT GOSUB
A8E2 2C	1318	.BY \$2C	;skip next instruction
A8E3 A211	1319	BA8E3 LDX \$11	;point to UNDEF'D STATEMENT message
A8E5 4C37A4	1320	JMP JA437	;go print error message
	1321	;	
A8E8 4C08AF	1322	BA8E8 JMP JAF08	;print SYNTAX Error message
	1323	;	
A8EB 68	1324	BA8EB PLA	;remove GOSUB block from stack
A8EC 68	1325	PLA	
A8ED 8539	1326	STA Z39	;remove statement number
A8EF 68	1327	PLA	
A8F0 853A	1328	STA Z3A	
A8F2 68	1329	PLA	
A8F3 857A	1330	STA Z7A	;restore current character pointer
A8F5 68	1331	PLA	
A8F6 857B	1332	STA Z7B	

```

1334 ;"DATA" command
1335 ;
A8F8 1336 WA8F8 = *
A8F8 2006A9 1337 JA8F8 JSR SA906 ;get offset to next ":"
A8FB 98 1338 BA8FB TYA
A8FC 18 1339 CLC
A8FD 657A 1340 ADC Z7A ;add offset to current character pointer
A8FF 857A 1341 STA Z7A ;(skips statement)
A901 9002 1342 BCC BA905
A903 E67B 1343 INC Z7B
A905 60 1344 BA905 RTS
1345 ;
1346 ;get end of statement (":" SA906)
1347 ;get end of line (00, SA909)
1348 ;
A906 A23A 1349 SA906 LDX `` ;set statement separator
A908 2C 1350 .BY $2C ;skip next instruction
A909 A200 1351 SA909 LDX $00 ;set line separator
A90B 8607 1352 STX Z07 ;save separator character
A90D A000 1353 LDY $00
A90F 8408 1354 STY Z08 ;save alternate separator
A911 A508 1355 BA911 LDA Z08
A913 A607 1356 LDX Z07
A915 8507 1357 STA Z07 ;swap separators
A917 8608 1358 STX Z08
A919 B17A 1359 BA919 LDA (Z7A),Y ;get next character
A91B F0E8 1360 BEQ BA905 ;if end of line
A91D C508 1361 CMP Z08 ;or separator reached
A91F F0E4 1362 BEQ BA905 ;exit with Y=length scanned
A921 C8 1363 INY ;else advance pointer
A922 C922 1364 CMP `` ;if character = quote
A924 D0F3 1365 BNE BA919 ;swap separators and continue
A926 F0E9 1366 BEQ BA911 ;continue
1367 ;
1368 ;"IF" command
1369 ;
A928 209EAD 1370 WA928 JSR SAD9E ;evaluate expression
A92B 207900 1371 JSR X0079 ;get current character
A92E C989 1372 CMP $89 ;if not code for GOTO
A930 F005 1373 BEQ BA937
A932 A9A7 1374 LDA $A7 ;or code for THEN
A934 20FFAE 1375 JSR SAEFF ;then SYNTAX Error
A937 A561 1376 BA937 LDA Z61 ;if result of condition true, do action
A939 D005 1377 BNE BA940 ;else skip by dropping through to REM

```


	1379	;	"REM" command	
	1380	;		
A93B 2009A9	1381	WA93B	JSR SA909	;get offset to end of line
A93E FOBB	1382		BEQ BA8FB	;go advance current character pointer
	1383	;		
	1384	;	THEN part of IF	
	1385	;		
A940 207900	1386	BA940	JSR X0079	;get current character
A943 B003	1387		BCS BA948	;if numeric
A945 4CA0A8	1388		JMP JA8A0	;then execute command GOTO
A948 4CEDA7	1389	BA948	JMP SA7ED	;else execute command in A
	1390	;		
	1391	;	"ON" command	
	1392	;		
A94B 209EB7	1393	WA94B	JSR SB79E	;get next integer value into X
A94E 48	1394		PHA	;save current character
A94F C98D	1395		CMP \$8D	;if not GOSUB
A951 F004	1396		BEQ BA957	
A953 C989	1397	BA953	CMP \$89	;or GOTO
A955 D091	1398		BNE BA8E8	;then SYNTAX Error
A957 C665	1399	BA957	DEC Z65	;decrement index
A959 D004	1400		BNE BA95F	;if zero
A95B 68	1401		PLA	;restore GOTO or GOSUB code in A
A95C 4CEFA7	1402		JMP JA7EF	;and execute command in A
A95F 207300	1403	BA95F	JSR X0073	;else get next character
A962 206BA9	1404		JSR SA96B	;gather decimal number into Zi4/Zi5
A965 C92C	1405		CMP ','	;if ends with a comma
A967 FOEE	1406		BEQ BA957	;then repeat
A969 68	1407		FLA	;else must be end of statement
A96A 60	1408	BA96A	RTS	;otherwise force SYNTAX Error

```

1410 ;gather decimal number into Z14/Z15
1411 ;
1412 ;gathered from current statement
1413 ;expects A, C and Z to reflect current character
1414 ;returns with A, C and Z reflecting new
1415 ;current character (first character after number)
1416 ;
A96B A200 1417 SA96B LDX $00 ;set answer area to zero
A96D 8614 1418 STX Z14
A96F 8615 1419 STX Z15
A971 B0F7 1420 JA971 BCS BA96A ;stop at first non-decimal character
A973 E92F 1421 SBC $2F ;convert decimal to binary
A975 8507 1422 STA Z07 ;save it
A977 A515 1423 LDA Z15
A979 8522 1424 STA Z22 ;save old high order
A97B C919 1425 CMP $19 ;if # gathered so far >= 6400
A97D B0D4 1426 BCS BA953 ;then SYNTAX Error
A97F A514 1427 LDA Z14
A981 0A 1428 ASL A ;* 2
A982 2622 1429 ROL Z22
A984 0A 1430 ASL A ;* 4
A985 2622 1431 ROL Z22
A987 6514 1432 ADC Z14 ;+ original = * 5
A989 8514 1433 STA Z14
A98B A522 1434 LDA Z22
A98D 6515 1435 ADC Z15
A98F 8515 1436 STA Z15
A991 0614 1437 ASL Z14 ;* 2 = * 10
A993 2615 1438 ROL Z15
A995 A514 1439 LDA Z14
A997 6507 1440 ADC Z07 ;add new digit
A999 8514 1441 STA Z14
A99B 9002 1442 BCC BA99F
A99D E615 1443 INC Z15
A99F 207300 1444 BA99F JSR X0073 ;get next character
A9A2 4C71A9 1445 JMP JA971 ;and repeat

```

```

1447 ;"LET" command
1448 ;
A9A5 1449 WA9A5 = *
A9A5 208BB0 1450 SA9A5 JSR SB08B ;gather name and get ptr to variable
A9A8 8549 1451 STA Z49 ;save variable pointer
A9AA 844A 1452 STY Z4A
A9AC A9B2 1453 LDA $B2 ;load code for "-"
A9AE 20FFAE 1454 JSR SAEFF ;must be next character
A9B1 A50E 1455 LDA Z0E
A9B3 48 1456 PHA ;save integer flag of variable
A9B4 A50D 1457 LDA Z0D
A9B6 48 1458 PHA ;save string flag of variable
A9B7 209EAD 1459 JSR SAD9E ;evaluate expression
A9BA 68 1460 PLA ;restore string flag of variable
A9BB 2A 1461 ROL A ;also into C flag
A9BC 2090AD 1462 JSR SAD90 ;check value for same type
A9BF D018 1463 BNE BA9D9 ;branch if string to be assigned
A9C1 68 1464 PLA ;restore integer flag of variable
A9C2 1012 1465 SA9C2 BPL BA9D6 ;branch if flp accu to be stored
1466 ;
1467 ;assign to integer
1468 ;
A9C4 201BBC 1469 JSR SBC1B ;round flp accu according to guard bit
A9C7 208FB1 1470 JSR SB1BF ;convert flp accu to integer
A9CA A000 1471 LDY $00
A9CC A564 1472 LDA Z64 ;get most significant byte of integer
A9CE 9149 1473 STA (Z49),Y ;store into + 0 of variable
A9D0 C8 1474 INY
A9D1 A565 1475 LDA Z65 ;get least significant byte of integer
A9D3 9149 1476 STA (Z49),Y ;store into + 1 of variable
A9D5 60 1477 RTS
1478 ;
1479 ;assign to flp
1480 ;
A9D6 4CDOB8 1481 BA9D6 JMP JBBD0 ;go store flp value
1482 ;
1483 ;assign to string
1484 ;
A9D9 68 1485 BA9D9 FLA ;remove integer flag from stack
A9DA A44A 1486 SA9DA LDY Z4A ;if high byte of pointer to variable
A9DC C0BF 1487 CPY $BF ;not same as address of dummy variable
A9DE D04C 1488 BNE BAA2C ;then go do normal variable

```

	1490	;assign to TI\$	
	1491	;	
A9E0	20A6B6	1492 JSR SB6A6	;de-allocate temporary string
A9E3	C906	1493 CMP \$06	;should be 6 characters long
A9E5	D03D	1494 BNE BAA24	;else ILLEGAL QUANTITY Error
A9E7	A000	1495 LDY \$00	
A9E9	8461	1496 STY Z61	;store 0 in flp accu
A9EB	8466	1497 STY Z66	
A9ED	8471	1498 BA9ED STY Z71	;set index in string
A9EF	201DAA	1499 JSR SAA1D	;add next digit to flp accu
A9F2	20E2BA	1500 JSR SBAE2	;multiply flp accu by 10
A9F5	E671	1501 INC Z71	;increment index
A9F7	A471	1502 LDY Z71	
A9F9	201DAA	1503 JSR SAA1D	;add next digit to flp accu
A9FC	200CBC	1504 JSR SBC0C	;move rounded flp accu to 2nd flp accu
A9FF	AA	1505 TAX	;get exponent
AA00	F005	1506 BEQ BAA07	;if zero, skip next multiplication
AA02	E8	1507 INX	;increment exponent
AA03	8A	1508 TXA	;*2 in second flp accu
AA04	20EDBA	1509 JSR SBAED	;add 2nd flp accu to flp accu and *2
AA07	A471	1510 BAA07 LDY Z71	;get index
AA09	C8	1511 INY	
AA0A	C006	1512 CPY \$06	;repeat until 6 characters processed
AA0C	D0DF	1513 BNE BA9ED	
AA0E	20E2BA	1514 JSR SBAE2	;multiply flp accu by 10
AA11	209BBC	1515 JSR SBC9B	;convert flp accu to 4 byte integer
AA14	A664	1516 LDX Z64	
AA16	A463	1517 LDY Z63	
AA18	A565	1518 LDA Z65	
AA1A	4CDBFF	1519 JMP XFFDB	;set real time clock
	1520	;	
	1521	;add next digit to flp accu	
	1522	;	
AA1D	B122	1523 SAA1D LDA (Z22),Y	;get character from string
AA1F	208000	1524 JSR X0080	;check for numeric character
AA22	9003	1525 BCC BAA27	
AA24	4C48B2	1526 BAA24 JMP JB248	;ILLEGAL QUANTITY Error if not
	1527	;	
AA27	E92F	1528 BAA27 SBC \$2F	;convert decimal to binary
AA29	4C7EED	1529 JMP JBD7E	;add signed integer from A to flp accu

```

1531 ;assign to normal string variable
1532 ;
AA2C A002 1533 BAA2C LDY $02
AA2E B164 1534 LDA (Z64),Y ;if pointer to string
AA30 C534 1535 CMP Z34
AA32 9017 1536 BCC BAA4B ;below allocated string area
AA34 D007 1537 BNE BAA3D
AA36 88 1538 DEY
AA37 B164 1539 LDA (Z64),Y
AA39 C533 1540 CMP Z33
AA3B 900E 1541 BCC BAA4B ;then move descriptor
AA3D A465 1542 BAA3D LDY Z65
AA3F C42E 1543 CPY Z2E
AA41 9008 1544 BCC BAA4B ;below end of program
AA43 D00D 1545 BNE BAA52 ;(descriptor stack)
AA45 A564 1546 LDA Z64
AA47 C52D 1547 CMP Z2D
AA49 B007 1548 BCS BAA52
AA4B A564 1549 BAA4B LDA Z64 ;then go move descriptor
AA4D A465 1550 LDY Z65
AA4F 4C68AA 1551 JMP JAA68
1552 ;
AA52 A000 1553 BAA52 LDY $00 ;else
AA54 B164 1554 LDA (Z64),Y ;get length of string
AA56 2075B4 1555 JSR SB475 ;allocate area
AA59 A550 1556 LDA Z50 ;store string pointer
AA5B A451 1557 LDY Z51
AA5D 856F 1558 STA Z6F
AA5F 8470 1559 STY Z70
AA61 207AB6 1560 JSR SB67A ;move string into allocated area
AA64 A961 1561 LDA <Z61 ;set AY to new descriptor
AA66 A000 1562 LDY >Z61
1563 ;
1564 ;move descriptor into variable
1565 ;
AA68 8550 1566 JAA68 STA Z50 ;set pointer to descriptor from AY
AA6A 8451 1567 STY Z51
AA6C 20DBB6 1568 JSR SB6DB ;check descriptor stack
AA6F A000 1569 LDY $00
AA71 B150 1570 LDA (Z50),Y ;move descriptor
AA73 9149 1571 STA (Z49),Y ;to variable
AA75 C8 1572 INY
AA76 B150 1573 LDA (Z50),Y
AA78 9149 1574 STA (Z49),Y ;also + 1
AA7A C8 1575 INY
AA7B B150 1576 LDA (Z50),Y
AA7D 9149 1577 STA (Z49),Y ;and + 2
AA7F 60 1578 RTS

```

```

1580 ;"PRINT#" command
1581 ;
AA80 2086AA 1582 WAA80 JSR SAA86 ;execute CMD and PRINT
AA83 4CB5AB 1583 JMP JABB5 ;reset CMD output file #
1584 ;
1585 ;"CMD" command
1586 ;
AA86 1587 WAA86 = *
AA86 209EB7 1588 SAA86 JSR SB79E ;get next integer value in X
AA89 F005 1589 BEQ BAA90 ;if more parameters
AA8B A92C 1590 LDA ',' ;next parameter must be a comma
AA8D 20FFAE 1591 JSR SAEFF
AA90 08 1592 BAA90 PHP ;save flags
AA91 8613 1593 STX Z13 ;store new CMD file #
AA93 2018E1 1594 JSR XE118 ;select output device
AA96 28 1595 PLP ;restore flags
AA97 4CA0AA 1596 JMP JAAA0 ;execute command PRINT
1597 ;
AA9A 2021AB 1598 BAA9A JSR SAB21 ;print BASIC string from Z22/Z23
AA9D 207900 1599 BAA9D JSR X0079 ;get current character
1600 ;
1601 ;"PRINT" command
1602 ;
AAA0 1603 WAAA0 = *
AAA0 F035 1604 JAAA0 BEQ SAAD7 ;if no parameters, end line on CMD file
AAA2 F043 1605 JAA2 BEQ BAAE7 ;if end of statement, return
AAA4 C9A3 1606 CMP $A3 ;if TAB(
AAA6 F050 1607 BEQ BAAF8 ;do TAB/SPC routine
AAA8 C9A6 1608 CMP $A6 ;if SPC(
AAAA 18 1609 CLC
AAAE F04B 1610 BEQ BAAF8 ;do TAB/SPC routine
AAD C92C 1611 CMP ',' ;if comma
AAAF F037 1612 BEQ BAAE8 ;go do fixed tab
AAB1 C93B 1613 CMP ';'; ;if semicolon
AAB3 F05E 1614 BEQ EAB13 ;ignore
AAB5 209EAD 1615 JSR SAD9E ;evaluate expression
AAB8 240D 1616 BIT Z0D ;if string
AABA 30DE 1617 BMI BAA9A ;print BASIC string
AABC 20DDBD 1618 JSR SBDDD ;convert flp accu to string
AABF 2087B4 1619 JSR SB487 ;get descriptor of string in flp accu
AAC2 2021AB 1620 JSR SAB21 ;print BASIC string from Z22/Z23
AAC5 203BAB 1621 JSR SAB3B ;print Cursor Right on CMD output file
AAC8 D0D3 1622 BNE BAA9D ;repeat

```

```

1624 ;end statement in buffer and on screen
1625 ;
1626 ;X=index beyond last byte in buffer
1627 ;
AACA A900 1628 JAACA LDA $00 ;add 00 to statement in buffer
AACC 9D0002 1629 STA X0200,X
AACF A2FF 1630 LDX $FF
AAD1 A001 1631 LDY $01 ;reset index to beginning of statement
AAD3 A513 1632 LDA Z13 ;if CMD output file = default
AAD5 D010 1633 BNE BAAE7 ;then end line on CMD output file
1634 ;
1635 ;end line on CMD output file
1636 ;
AAD7 A90D 1637 SAAD7 LDA $0D ;print a Return
AAD9 2047AB 1638 JSR SAB47 ;on CMD output file
AADC 2413 1639 BIT Z13 ;if file number < 128
AADE 1005 1640 BPL BAAE5 ;do not add Linefeed
AAE0 A90A 1641 LDA $0A ;else print a Linefeed on CMD file
AAE2 2047AB 1642 JSR SAB47
AAE5 49FF 1643 BAAE5 EOR $FF ;condition flags
AAE7 60 1644 BAAE7 RTS
1645 ;
1646 ;routine for printing blanks and tabs
1647 ;
1648 ;fixed tabs at BAAE8
1649 ;TAB( & SPC( at BAAF8
1650 ;
AAE8 38 1651 BAAE8 SEC
AAE9 20FOFF 1652 JSR XFFFO ;read cursor position into XY
AAEC 98 1653 TYA
AAED 38 1654 SEC
AAEE E90A 1655 BAAEE SBC $0A ;convert column to modulo 10 (~10 to -1)
AAF0 B0FC 1656 BCS BAAEE
AAF2 49FF 1657 EOR $FF ;complement
AAF4 6901 1658 ADC $01 ;A = # blanks to insert
AAF6 D016 1659 BNE BABOE ;skip if zero
AAF8 08 1660 BAAF8 PHP ;save status
AAF9 38 1661 SEC
AAFA 20FOFF 1662 JSR XFFFO ;read cursor position again
AAFD 8409 1663 STY Z09 ;save column
AAFF 209BB7 1664 JSR SB79B ;evaluate expression into X
AB02 C929 1665 CMP ") ;if not ending with ")"
AB04 D059 1666 BNE BAB5F ;then SYNTAX Error
AB06 28 1667 PLP ;restore status
AB07 9006 1668 BCC BAB0F ;if code for TAB(
AB09 8A 1669 TXA ;get value in A
AB0A E509 1670 SBC Z09 ;subtract current character position
AB0C 9005 1671 BCC BAB13 ;skip if negative
AB0E AA 1672 BABOE TAX
AB0F E8 1673 BAB0F INX
AB10 CA 1674 BAB10 DEX
AB11 D006 1675 BNE BAB19 ;if zero
AB13 207300 1676 BAB13 JSR X0073 ;get next character
AB16 4CA2AA 1677 JMP JAAA2 ;return to PRINT
AB19 203BAB 1678 BAB19 JSR SAB3B ;print Cursor Right on CMD output file
AB1C D0F2 1679 BNE BAB10 ;X times

```

```

1681 ;print string from AY
1682 ;
AB1E 2087B4 1683 SAB1E JSR SB487 ;move string descriptor into flip accu
1684 ;
1685 ;print BASIC string from Z22/Z23
1686 ;
AB21 20A6B6 1687 SAB21 JSR SB6A6 ;get text pointer into Z22/Z23
AB24 AA 1688 TAX ;save string length
AB25 A000 1689 LDY $00
AB27 E2 1690 INX
AB28 CA 1691 BAB28 DEX ;decrement length pending
AB29 F0BC 1692 BEQ BAAE7 ;exit upon completion
AB2B B122 1693 LDA (Z22),Y ;get next character
AB2D 2047AB 1694 JSR SAB47 ;print it on CMD output file
AB30 C8 1695 INY ;advance pointer
AB31 C90D 1696 CMP $0D ;if character printed was a Return
AB33 D0F3 1697 BNE BAB28
AB35 20E5AA 1698 JSR BAAE5 ;then do dummy instruction
AB38 4C28AB 1699 JMP BAB28 ;repeat
1700 ;
1701 ;print a character on CMD output file
1702 ;
AB3B A513 1703 SAB3B LDA Z13 ;if CMD output file is default
AB3D F003 1704 BEQ BAB42 ;print Cursor Right
AB3F A920 1705 LDA $20 ;else use space
AB41 2C 1706 .BY $2C ;skip next instruction
AB42 A91D 1707 BAB42 LDA $1D ;load Cursor Right
AB44 2C 1708 .BY $2C ;skip next instruction
AB45 A93F 1709 SAB45 LDA $3F ;load "?"
AB47 200CE1 1710 SAB47 JSR XE10C ;output a character
AB4A 29FF 1711 AND $FF ;and exit with flags corresponding
AB4C 60 1712 RTS ;to original A

```


	1714	;read errors	
	1715	;	
AB4D A511	1716	JAB4D LDA Z11	
AB4F F011	1717	BEQ BAB62	;skip if INPUT
AB51 3004	1718	BMI BAB57	;if GET
AB53 A0FF	1719	LDY SFF	;set statement # invalid
AB55 D004	1720	BNE BAB5B	
AB57 A53F	1721	BAB57 LDA Z3F	;for READ
AB59 A440	1722	LDY Z40	;get DATA statement #
AB5B 8539	1723	BAB5B STA Z39	;store statement #
AB5D 843A	1724	STY Z3A	
AB5F 4C08AF	1725	BAB5F JMP JAF08	;SYNTAX Error
AB62 A513	1726	BAB62 LDA Z13	;if INPUT and not default CMD file
AB64 F005	1727	BEQ BAB6B	
AB66 A218	1728	LDX \$18	;point to FILE DATA message
AB68 4C37A4	1729	JMP JA437	;print error
	1730	;	
AB6B A90C	1731	BAB6B LDA <TADOC	;if INPUT and default CMD output file
AB6D A0AD	1732	LDY >TADOC	;index REDO FROM START message
AB6F 201EAB	1733	JSR SAB1E	;print message
AB72 A53D	1734	LDA Z3D	;move pointer to current statement
AB74 A43E	1735	LDY Z3E	
AB76 857A	1736	STA Z7A	;into current character pointer
AB78 847B	1737	STY Z7B	
AB7A 6C	1738	RTS	
	1739	;	
	1740	;"GET" command	
	1741	;	
AB7B 20A6B3	1742	WAB7B JSR SB3A6	;check for direct mode
AB7E C923	1743	CMP "#	;if next character is # sign
AB80 D010	1744	BNE BAB92	
AB82 207300	1745	JSR X0073	;then get next character
AB85 209EB7	1746	JSR SB79E	;and next integer value in X
AB88 A92C	1747	LDA "	
AB8A 20FFAE	1748	JSR SAEFF	;next character must be a comma
AB8D 8613	1749	STX Z13	;set new CMD input file
AB8F 201EE1	1750	JSR XE11E	;open device for input
AB92 A201	1751	BAB92 LDX \$01	;set pointer to end of line
AB94 A002	1752	LDY \$02	
AB96 A900	1753	LDA \$00	
AB98 8D0102	1754	STA X0200+1	;set character string terminator
AB9B A940	1755	LDA \$40	;set code for GET
AB9D 200FAC	1756	JSR SACOF	;interpret data
ABAO A613	1757	LDX Z13	;get CMD output file
ABA2 D013	1758	BNE BABB7	;if not default go reset it
ABA4 60	1759	RTS	
	1760	;	
	1761	;"INPUT#" command	
	1762	;	
ABA5 209EB7	1763	WABA5 JSR SB79E	;get next integer value in X
ABA8 A92C	1764	LDA "	
ABAA 20FFAE	1765	JSR SAEFF	;next character must be a comma
ABAD 8613	1766	STX Z13	;set CMD file #
ABAF 201EE1	1767	JSR XE11E	;open device for input
ABB2 20CEAB	1768	JSR SABCE	;do Input
ABB5 A513	1769	JABB5 LDA Z13	;get CMD file number
ABB7 20CCFF	1770	BABB7 JSR XFFCC	;deselect all devices
ABBA A200	1771	LDX \$00	
ABBC 8613	1772	STX Z13	;set CMD file to default
ABBE 60	1773	RTS	

```

1775 ;"INPUT" command
1776 ;
ABBF C922 1777 WABBF CMP "" ;if next character is a quote
ABC1 D00B 1778 BNE SABCE
ABC3 20BDAE 1779 JSR SAEBD ;get descriptor of literal prompt string
ABC6 A93B 1780 LDA " ;if next character is not a ";"
ABCE 20FFAE 1781 JSR SAEFF
ABCB 2021AB 1782 JSR SAB21 ;then SYNTAX Error
ABCE 20A6B3 1783 SABCE JSR SB3A6 ;print prompt string
ABD1 A92C 1784 LDA " ;place a comma before input string
ABD3 8DFF01 1785 STA X01FF
ABD6 20F9AB 1786 BABD6 JSR SABF9 ;get input line into input buffer
ABD9 A513 1787 LDA Z13 ;if CMD file
ABDB F00D 1788 BEQ BABEA ;is not default
ABDD 20B7FF 1789 JSR XFFB7 ;and ST does not show
ABE0 2902 1790 AND $02 ;a Read Error
ABE2 F006 1791 BEQ BABEA
ABE4 20B5AB 1792 JSR JABB5 ;reset CMD file to default
ABE7 4CF8AB 1793 JMP JABF8 ;execute command DATA
1794 ;
ABEA AD0002 1795 BABEA LDA X0200 ;else if not a null line
ABED D01E 1796 BNE BACOD ;interpret data
ABEF A513 1797 LDA Z13 ;if null line and CMD file not default
ABF1 D0E3 1798 BNE BABD6 ;skip null line
ABF3 2006A9 1799 JSR SA906 ;get end of line
ABF6 4CFBAA 1800 JMP BABF8 ;skip rest of statement
1801 ;
1802 ;get line into input buffer
1803 ;
ABF9 A513 1804 SABF9 LDA Z13 ;if CMD file = default
ABFB D006 1805 BNE BAC03
ABFD 2045AB 1806 JSR SAB45 ;print "?" on CMD file
AC00 203BAB 1807 JSR SAB3B ;print Cursor Right on CMD file
AC03 4C60A5 1808 BAC03 JMP SA560 ;go get statement into input buffer
1809 ;
1810 ;"READ" command
1811 ;
AC06 A641 1812 WAC06 LDX Z41 ;get current DATA address into XY
AC08 A442 1813 LDY Z42
AC0A A998 1814 LDA $98 ;set READ code
AC0C 2C 1815 .BY $2C ;skip next instruction
AC0D A900 1816 BAC0D LDA $00 ;for INPUT, set A to 0
AC0F 8511 1817 SAC0F STA Z11 ;save entry mode
AC11 8643 1818 STX Z43 ;save XY into temporary read pointer
AC13 8444 1819 STY Z44
AC15 208BB0 1820 JAC15 JSR SB08B ;gather name & get pointer to variable
AC18 8549 1821 STA Z49 ;store pointer
AC1A 844A 1822 STY Z4A
AC1C A57A 1823 LDA Z7A ;get current character pointer
AC1E A47B 1824 LDY Z7B
AC20 854B 1825 STA Z4B ;save it
AC22 844C 1826 STY Z4C
AC24 A643 1827 LDX Z43 ;get temporary read pointer
AC26 A444 1828 LDY Z44
AC28 867A 1829 STX Z7A ;use it as current character pointer
AC2A 847B 1830 STY Z7B
AC2C 207900 1831 JSR X0079 ;get current character
AC2F D020 1832 BNE BAC51 ;if end of statement, get next character
AC31 2411 1833 BIT Z11 ;if GET mode
AC33 500C 1834 BVC BAC41

```

AC35	2024E1	1835	JSR	XE124		;get a character from CMD file
AC38	8D0002	1836	STA	X0200		;save character in input buffer
AC3B	A2FF	1837	LDX	\$FF		
AC3D	A001	1838	LDY	\$01		;move buffer address - 1 into XY
AC3F	D00C	1839	BNE	BAC4D		
AC41	3075	1840	BAC41	BMI	BACB8	;if READ go get next DATA element
AC43	A513	1841	LDA	Z13		;for INPUT if CMD file = default
AC45	D003	1842	BNE	BAC4A		
AC47	2045AB	1843	JSR	SAB45		;print "?" on CMD file
AC4A	20F9AB	1844	BAC4A	JSR	SABF9	;get new input line
AC4D	867A	1845	BAC4D	STX	Z7A	;store pointer into current pointer
AC4F	847B	1846		STY	Z7B	
AC51	207300	1847	BAC51	JSR	X0073	;get next pointer
AC54	240D	1848		BIT	Z0D	;if not a string variable
AC56	1031	1849		BPL	BAC89	
AC58	2411	1850		BIT	Z11	;and GET mode
AC5A	5009	1851		BVC	BAC65	
AC5C	E8	1852		INX		
AC5D	867A	1853		STX	Z7A	;skip character
AC5F	A900	1854		LDA	\$00	;set null as terminator
AC61	8507	1855		STA	Z07	
AC63	F00C	1856		BEQ	BAC71	;if string variable and READ or INPUT
AC65	8507	1857	BAC65	STA	Z07	;store character as terminator
AC67	C922	1858		CMP	" "	;if not quote character
AC69	F007	1859		BEQ	BAC72	;store ":" as terminator
AC6B	A93A	1860		LDA	:	
AC6D	8507	1861		STA	Z07	
AC6F	A92C	1862		LDA	,	
AC71	18	1863	BAC71	CLC		
AC72	8508	1864	BAC72	STA	Z08	;and ",", as an alternative terminator
AC74	A57A	1865		LDA	Z7A	
AC76	A47B	1866		LDY	Z7B	
AC78	6900	1867		ADC	\$00	;skip initial quote if any
AC7A	9001	1868		BCC	BAC7D	
AC7C	C8	1869		INY		
AC7D	208DB4	1870	BAC7D	JSR	SB48D	;get descriptor of string into flp accu
AC80	20E2B7	1871		JSR	SB7E2	;move text pointer into current pointer
AC83	20DAA9	1872		JSR	SA9DA	;assign to string
AC86	4C91AC	1873		JMP	JAC91	
		1874				
AC89	20F3BC	1875	BAC89	JSR	SBCF3	;convert string into flp accu
AC8C	A50E	1876		LDA	Z0E	;load integer flag
AC8E	20C2A9	1877		JSR	SA9C2	;assign to integer or real
AC91	207900	1878	JAC91	JSR	X0079	;get current character
AC94	F558	1879		FTK	F:A-U	Xjqfh ca ael rp fhihajsah
AC96	C92C	1880		CMP	,	;or a comma
AC98	F003	1881		BEQ	BAC9D	
AC9A	4C4DAB	1882		JMP	JAB4D	;else read error
		1883				
AC9D	A57A	1884	BAC9D	LDA	Z7A	;move current character pointer
AC9F	A47B	1885		LDY	Z7B	
ACA1	8543	1886		STA	Z43	;into temporary read pointer
ACA3	8444	1887		STY	Z44	
ACA5	A54B	1888		LDA	Z4B	;move saved current character pointer
ACA7	A44C	1889		LDY	Z4C	
ACA9	857A	1890		STA	Z7A	;to current character pointer
ACAB	847B	1891		STY	Z7B	
ACAD	207900	1892		JSR	X0079	;get current character
ACB0	F02D	1893		BEQ	BACDF	;must be end of statement
ACB2	20FDAE	1894		JSR	SAEFD	;or a comma

```

ACB5 4C15AC 1895      JMP JAC15      ;if so repeat read
                        1896 ;
ACB8 2006A9 1897 BACB8 JSR SA906      ;if end of line during READ, get index
ACB8 C8      1898      INY            ;to end of statement and bump it
ACB8 AA      1899      TAX            ;move last character
ACB8 D012    1900      BNE BACD1      ;if zero,
ACB8 A20D    1901      LDX $0D        ;point to OUT OF DATA message
ACC1 C8      1902      INY
ACC2 B17A    1903      LDA (Z7A),Y    ;if high of ptr to next statement zero
ACC4 F06C    1904      BEQ BAD32      ;fatal error
ACC6 C8      1905      INY
ACC7 B17A    1906      LDA (Z7A),Y    ;move low byte of next statement #
ACC9 853F    1907      STA Z3F        ;into DATA statement #
ACCB C8      1908      INY
ACCC B17A    1909      LDA (Z7A),Y    ;high byte also
ACCE C8      1910      INY
ACCF 8540    1911      STA Z40
ACD1 20FBA8 1912 BACD1 JSR BA8FB      ;add offset to current character pointer
ACD4 207900 1913      JSR X0079      ;get current character
ACD7 AA      1914      TAX
ACD8 E083    1915      CFX $83        ;if not code for DATA
ACDA D0DC    1916      BNE BACE8      ;go skip statement
ACDC 4C51AC 1917      JMP BAC51      ;set READ pointer
                        1918 ;
ACDF A543    1919 BACDF LDA Z43        ;at end of READ statement
ACE1 A444    1920      LDY Z44        ;get temporary read pointer
ACE3 A611    1921      LDX Z11        ;if GET or READ mode
ACE5 1003    1922      BPL BACEA
ACE7 4C27A8 1923      JMP BA827      ;go set READ pointer
                        1924 ;
ACEA A000    1925 BACEA LDY $00        ;if INPUT
ACEC B143    1926      LDA (Z43),Y    ;must be end of line
ACEE F00B    1927      BEQ BACFB
ACF0 A513    1928      LDA Z13        ;else, if CMD file = default
ACF2 D007    1929      BNE BACFB
ACF4 A9FC    1930      LDA <TACFC     ;set AY to message EXRTA IGNORED
ACF6 A0AC    1931      LDY >TACFC
ACF8 4C1EAB 1932      JMP SAB1E      ;print message
                        1933 ;
ACFB 60      1934 BACFB RTS
                        1935 ;
                        1936 ;messages used during READ
                        1937 ;
ACFC 3F4558 1938 TACFC .BY $3F,"E","X","T","R","A"," ","I","G","N","O","R","E","D,$0D,$00
AD0C 3F5245 1939 TAD0C .BY $3F,"R","E","D","O"," ","F","R","O","M"," ","S","T","A","R","T
AD1C OD00    1940      .BY $0D,$00

```

```

1942 ;"NEXT" command
1943 ;
AD1E D004 1944 WAD1E BNE SAD24 ;if no parameters
AD20 A000 1945 LDY $00 ;set index pointer to variable
AD22 F003 1946 BEQ BAD27
AD24 208BB0 1947 SAD24 JSR SB08B ;else gather name & get ptr to variable
AD27 8549 1948 BAD27 STA Z49 ;save pointer
AD29 844A 1949 STY Z4A
AD2B 208AA3 1950 JSR SA38A ;get corresponding FOR block
AD2E F005 1951 BEQ BAD35
AD30 A20A 1952 LDX $0A ;if not found,
AD32 4C37A4 1953 BAD32 JMP JA437 ;error NEXT WITHOUT FOR
1954 ;
AD35 9A 1955 BAD35 TXS ;if found remove everything on top
AD36 8A 1956 TXA
AD37 18 1957 CLC
AD38 6904 1958 ADC $04
AD3A 48 1959 PHA ;get stack index of STEP value (+ 3)
AD3B 6906 1960 ADC $06
AD3D 8524 1961 STA Z24
AD3F 68 1962 PLA ;save stack index of TO value (+ 9)
AD40 A001 1963 LDY $01 ;high order byte of stack pointer
AD42 20A2BB 1964 JSR SBBA2 ;restore flp accu
AD45 BA 1965 TSX ;get top index
AD46 BD0901 1966 LDA X0100+9,X ;get sign of STEP value (+ 8)
AD49 8566 1967 STA Z66 ;restore sign byte of flp accu
AD4B A549 1968 LDA Z49
AD4D A44A 1969 LDY Z4A ;AY points to FOR variable
AD4F 2067B8 1970 JSR SB867 ;add flp accu to # at AY
AD52 20D0BB 1971 JSR JBBDO ;store flp accu into FOR variable
AD55 A001 1972 LDY $01
AD57 205DBC 1973 JSR SBC5D ;compare flp accu to upper limit
AD5A BA 1974 TSX ;get top index
AD5B 38 1975 SEC
AD5C FD0901 1976 SBC X0100+9,X ;if different from sign of STEP
AD5F F017 1977 BEQ BAD78
AD61 BDOF01 1978 LDA X0100+15,X
AD64 8539 1979 STA Z39 ;restore statement # to start of loop
AD66 BD1001 1980 LDA X0100+16,X
AD69 853A 1981 STA Z3A
AD6B BD1201 1982 LDA X0100+18,X
AD6E 857A 1983 STA Z7A ;restore character ptr to start of loop
AD70 BD1101 1984 LDA X0100+17,X
AD73 857B 1985 STA Z7B
AD75 4CAEA7 1986 BAD75 JMP JA7AE ;go execute next statement
1987 ;
AD78 8A 1988 BAD78 TXA
AD79 6911 1989 ADC $11 ;add 17 to stack pointer
AD7B AA 1990 TAX ;to remove FOR block
AD7C 9A 1991 TXS
AD7D 207900 1992 JSR X0079 ;get current character
AD80 C92C 1993 CMP ',' ;if a comma then execute next statement
AD82 D0F1 1994 BNE BAD75
AD84 207300 1995 JSR X0073 ;else get next character
AD87 2024AD 1996 JSR SAD24 ;and go repeat NEXT statement

```

```

1998 ;get next non-string value
1999 ;
AD8A 209EAD 2000 SAD8A JSR SAD9E ;get next value
2001 ;
AD8D 18 2002 SAD8D CLC ;set string not wanted
AD8E 24 2003 .BY $24 ;skip next instruction
2004 ;
2005 ;check value to be string
2006 ;
AD8F 38 2007 SAD8F SEC ;set string wanted
2008 ;
2009 ;check value according to C flag
2010 ;
AD90 240D 2011 SAD90 BIT ZOD ;test string flag
AD92 3003 2012 BMI BAD97 ;if not string
AD94 B003 2013 BCS BAD99 ;and string wanted, error
AD96 60 2014 BAD96 RTS ;else return
2015 ;
AD97 B0FD 2016 BAD97 BCS BAD96 ;if string and string wanted, return
AD99 A216 2017 BAD99 LDX $16 ;if string and string not wanted, index
AD9B 4C37A4 2018 JMP JA437 ;TYPE MISMATCH and print error
2019 ;
2020 ;evaluate expression
2021 ;
AD9E A67A 2022 SAD9E LDX Z7A ;decrement character pointer
ADA0 D002 2023 BNE BADA4
ADA2 C67B 2024 DEC Z7B
ADA4 C67A 2025 BADA4 DEC Z7A
ADA6 A200 2026 LDX $00 ;initial priority
ADA8 24 2027 .BY $24 ;skip next instruction
ADA9 48 2028 JADA9 PHA ;save <=> code
ADAA 8A 2029 TXA
ADAB 48 2030 PHA ;save priority
ADAC A901 2031 LDA $01 ;one slot on stack wanted
ADAE 20FBA3 2032 JSR SAE ;else error OUT OF MEMORY Error
ADB1 2083AE 2033 JSR SAE8 ;get value of next operand in flp accu
ADB4 A900 2034 LDA $00
ADB6 854D 2035 STA Z4D ;initial <=> code for next diadic
ADB8 207900 2036 JADB8 JSR X0079 ;get current character
ADBB 38 2037 JADBB SEC
ADBC E9B1 2038 SBC $B1 ;subtract code for >
ADBE 9017 2039 BCC BADD7 ;branch if lower than >
ADC0 C903 2040 CMP $03
ADC2 B013 2041 BCS BADD7 ;or if higher than <
ADC4 C901 2042 CMP $01
ADC6 2A 2043 ROL A ;compute bits for <=>
ADC7 4901 2044 EOR $01 ;bit 2 for <, 1 for = and 0 for >
ADC9 454D 2045 EOR Z4D ;add to existing <=> code
ADCB C54D 2046 CMP Z4D ;if bit added for the second time
ADCD 9061 2047 BCC BAE30 ;then SYNTAX Error
ADCF 854D 2048 STA Z4D
ADD1 207300 2049 JSR X0073 ;get next character
ADD4 4CBBAD 2050 JMP JADBB ;and repeat
2051 ;
ADD7 A64D 2052 BADD7 LDX Z4D
ADD9 D02C 2053 BNE BAE07 ;if no <=> code
ADDB B07B 2054 BCS BAE58 ;then end at non-diadic character
ADD 6907 2055 ADC $07 ;diadic index 0-6
ADDF 9077 2056 BCC BAE58
ADE1 650D 2057 ADC ZOD ;add string flag + carry

```

```

ADE3 D003 2058 BNE BADE8 ;if still zero
ADE5 4C3DB6 2059 JMP JB63D ;apply diadic operator "+" on strings
2060 ;
ADE8 69FF 2061 BADE8 ADC $FF ;make index 0-6 again
ADEA 8522 2062 STA Z22
ADEC 0A 2063 ASL A
ADED 6522 2064 ADC Z22 ;* 3
ADEF A8 2065 TAY
ADF0 68 2066 BADF0 PLA ;restore priority of stacked diadic
ADF1 D980A0 2067 CMP TA080,Y ;compare to priority of current diadic
ADF4 B067 2068 BCS BAE5D ;if stacked higher, apply operator
ADF6 208DAD 2069 JSR SAD8D ;TYPE MISMATCH if string operand
ADF9 48 2070 BADF9 PHA ;save priority again
ADFA 2020AE 2071 JADFA JSR SAE20 ;recursive call to evaluate expression
ADFD 68 2072 PLA ;restore priority again
ADFE A44B 2073 LDY Z4B ;restore index of current diadic
AE00 1017 2074 BPL BAE19 ;if at end
AE02 AA 2075 TAX ;and priority = 0
AE03 F056 2076 BEQ BAE5B ;stop
AE05 D05F 2077 BNE BAE66 ;else apply
AE07 460D 2078 BAE07 LSR Z0D ;get string flag
AE09 8A 2079 TXA
AE0A 2A 2080 ROL A ;add to <=> code
AE0B A67A 2081 LDX Z7A
AE0D D002 2082 BNE BAE11 ;decrement character pointer
AE0F C67B 2083 DEC Z7B
AE11 C67A 2084 BAE11 DEC Z7A
AE13 A01B 2085 LDY $1B ;get index (3*9)
AE15 854D 2086 STA Z4D ;save <=> code
AE17 D0D7 2087 BNE BADF0 ;and go check priorities
AE19 D980A0 2088 BAE19 CMP TA080,Y ;if true diadic, compare priorities
AE1C B048 2089 BCS BAE66 ;if stacked priority higher, apply it
AE1E 90D9 2090 BCC BADF9 ;else repeat
2091 ;
2092 ;recursive entry for evaluation of expressions
2093 ;
AE20 B982A0 2094 SAE20 LDA TA080+2,Y ;save pointer to routine for diadic
AE23 48 2095 PHA
AE24 B981A0 2096 LDA TA080+1,Y
AE27 48 2097 PHA
AE28 2033AE 2098 JSR SAE33 ;save rounded value of left operand
AE2B A54D 2099 LDA Z4D ;get <=> code
AE2D 4CA9AD 2100 JMP JADA9 ;and restart evaluation
2101 ;
AE30 4C08AF 2102 BAE30 JMP JAF08 ;print SYNTAX Error
2103 ;
2104 ;save rounded value of left operand
2105 ;
AE33 A566 2106 SAE33 LDA Z66 ;get sign of flp accu
AE35 BE80A0 2107 LDX TA080,Y ;get priority of diadic
AE38 A8 2108 SAE38 TAY
AE39 68 2109 PLA
AE3A 8522 2110 STA Z22 ;save return address in Z22/Z23
AE3C E622 2111 INC Z22
AE3E 68 2112 PLA
AE3F 8523 2113 STA Z23
AE41 98 2114 TYA
AE42 48 2115 PHA ;save sign of flp accu
AE43 201BBC 2116 JAE43 JSR SBC1B ;round flp accu
AE46 A565 2117 LDA Z65 ;save flp accu on stack

```

AE48	48	2118	PHA	
AE49	A564	2119	LDA Z64	
AE4B	48	2120	PHA	
AE4C	A563	2121	LDA Z63	
AE4E	48	2122	PHA	
AE4F	A562	2123	LDA Z62	
AE51	48	2124	PHA	
AE52	A561	2125	LDA Z61	
AE54	48	2126	PHA	
AE55	6C2200	2127	JMP (Z22)	;return to caller
		2128	;	
		2129	;apply diadic operator	
		2130	;	
AE58	A0FF	2131	BAE58 LDY \$FF	;at expression end, set negative index
AE5A	68	2132	PLA	;restore diadic priority from stack
AE5B	F023	2133	BAE5B BEQ BAE80	;if zero, stop
AE5D	C964	2134	BAE5D CMP \$64	;if not priority of <=>
AE5F	F003	2135	BEQ BAE64	
AE61	208DAD	2136	JSR SAD8D	;then TYPE MISMATCH if string
AE64	844B	2137	BAE64 STY Z4B	;save index of current diadic
AE66	68	2138	BAE66 PLA	;remove string flag
AE67	4A	2139	LSR A	
AE68	8512	2140	STA Z12	;save <=> code for <=> routine
AE6A	68	2141	PLA	;restore left operand
AE6B	8569	2142	STA Z69	;to second flip accu
AE6D	68	2143	PLA	
AE6E	856A	2144	STA Z6A	
AE70	68	2145	PLA	
AE71	856B	2146	STA Z6B	
AE73	68	2147	PLA	
AE74	856C	2148	STA Z6C	
AE76	68	2149	PLA	
AE77	856D	2150	STA Z6D	
AE79	68	2151	PLA	
AE7A	856E	2152	STA Z6E	
AE7C	4566	2153	EOR Z66	;Exclusive OR of both signs
AE7E	856F	2154	STA Z6F	
AE80	A561	2155	BAE80 LDA Z61	;return with A = exponent of result
AE82	60	2156	RTS	


```

AE83 6C0A03 2158 SAE83 JMP (X030A) ;get arithmetic element (normally AE86)
                2159 ;
                2160 ;get arithmetic element into flp accu
                2161 ;
AE86 A900 2162 LDA $00 ;set string flag to not string
AE88 850D 2163 STA Z0D
AE8A 207300 2164 BAE8A JSR X0073 ;get next character
AE8D B003 2165 BCS BAE92 ;if numeric,
AE8F 4CF3BC 2166 BAE8F JMP SBCF3 ;gather number
                2167 ;
AE92 2013B1 2168 BAE92 JSR SB113 ;check for alphabetic character
AE95 9005 2169 BCC BAE9A
AE97 4C28AF 2170 JMP JAF28 ;if so, get value for variable
                2171 ;
AE9A C9FF 2172 BAE9A CMP $FF ;if code for PI
AE9C D00F 2173 BNE BAEAD
AE9E A9A8 2174 LDA <TAEAS ;let AY point to value for PI
AEA0 A0AE 2175 LDY >TAEAS
AEA2 20A2BB 2176 JSR SBBA2 ;load value into flp accu
AEA5 4C7300 2177 JMP X0073 ;get next character
                2178 ;
                2179 ;flp value for PI
                2180 ;
AEAS 82490F 2181 TAEAS .BY $82,$49,$0F,$DA,$A1
                2182 ;
AEAD C92E 2183 BAEAD CMP " ;if decimal point
AEAF F0DE 2184 BEQ BAE8F ;gather number
AEB1 C9AB 2185 CMP $AB ;if minus sign
AEB3 F058 2186 BEQ BAF0D ;do recursive get value
AEB5 C9AA 2187 CMP $AA ;if plus
AEB7 F0D1 2188 BEQ BAE8A ;ignore
AEB9 C922 2189 CMP "" ;if quote
AEBB D00F 2190 BNE BAEC6
AEBD A57A 2191 SAEBD LDA Z7A ;get pointer to first char of string
AEBF A47B 2192 LDY Z7B
AEC1 6900 2193 ADC $00 ;into Y
AEC3 9001 2194 BCC BAEC6
AEC5 C8 2195 INY
AEC6 2087B4 2196 BAEC6 JSR SB487 ;get ptr to desc of constant string
AEC9 4CE2B7 2197 JMP SB7E2 ;set current ptr after string
                2198 ;
AECC C9A8 2199 BAEC6 CMP $A8 ;if NOT
AECE D013 2200 BNE BAEE3
AEDO A018 2201 LDY $18 ;set index for NOT
AED2 D03B 2202 BNE BAF0F ;do recursive get value

```

```

2204 ;monadic "NOT" command
2205 ;
AED4 20BFB1 2206 WAED4 JSR SB1BF ;convert flp to integer
AED7 A565 2207 LDA Z65
AED9 49FF 2208 EOR $FF ;get complement of low byte into A
AEDB A8 2209 TAY
AEDC A564 2210 LDA Z64 ;get complement of high byte in A
AEDE 49FF 2211 EOR $FF
AEE0 4C91B3 2212 JMP JB391 ;convert integer to flp
2213 ;
2214 ;continuation of GET operand
2215 ;
AEE3 C9A5 2216 BAE3 CMP $A5 ;if FN
AEE5 D003 2217 BNE BAE4
AEE7 4CF4B3 2218 JMP JB3F4 ;expand function
2219 ;
AEEA C9B4 2220 BAE4 CMP $B4 ;if code for SGN or higher
AEEC 9003 2221 BCC BAEF1
AEEE 4CA7AF 2222 JMP JAF47 ;go apply function
2223 ;
AEF1 20FAAE 2224 BAEF1 JSR SAEFA ;SYNTAX Error if no "(" present
AEF4 209EAD 2225 JSR SAD9E ;evaluate expression
2226 ;
2227 ;check and skip characters
2228 ;
AEF7 A929 2229 SAEF7 LDA "(" ;check for ")"
AEF9 2C 2230 .BY $2C ;skip next instruction
AEFA A928 2231 SAEFA LDA "(" ;check for "("
AEFC 2C 2232 .BY $2C ;skip next instruction
AEFD A92C 2233 SAEFD LDA "," ;check for ","
AEFF A000 2234 SAEFF LDY $00 ;check for code in A
AF01 D17A 2235 CMP (27A),Y ;if current character < > to parameter
AF03 D003 2236 BNE JAF08 ;then SYNTAX Error
AF05 4C7300 2237 JMP X0073 ;else get next character and return
2238 ;
AF08 A20B 2239 JAF08 LDX $0B ;point to SYNTAX Error message
AFOA 4C37A4 2240 JMP JA437 ;print message
2241 ;
2242 ;recursive get value
2243 ;
AF0D A015 2244 BAF0D LDY $15 ;set index for "--"
AF0F 68 2245 BAF0F PLA ;remove own return address
AF10 68 2246 PLA
AF11 4CFAAD 2247 JMP JADFA ;do pseudo diadic
2248 ;
2249 ;check variable pointer range
2250 ;
AF14 38 2251 SAF14 SEC
AF15 A564 2252 LDA Z64
AF17 E900 2253 SBC $00
AF19 A565 2254 LDA Z65
AF1B E9A0 2255 SBC $A0
AF1D 9008 2256 BCC BAF27 ;exit with C clear if pointer < $A000
AF1F A9A2 2257 LDA $A2
AF21 E564 2258 SBC Z64
AF23 A9E3 2259 LDA $E3
AF25 E565 2260 SBC Z65 ;clear C if $E342 < pointer
AF27 60 2261 BAF27 RTS

```

```

2263 ;get value of variable
2264 ;
AF28 2088B0 2265 JAF28 JSR SB08B ;gather name and get pointer to variable
AF2B 8564 2266 STA Z64 ;save pointer to variable
AF2D 8465 2267 STY Z65
AF2F A645 2268 LDX Z45 ;get name of variable
AF31 A446 2269 LDY Z46
AF33 A50D 2270 LDA Z0D ;if not a string
AF35 F026 2271 BEQ BAF5D ;go check for integer and flp
AF37 A900 2272 LDA $00
AF39 8570 2273 STA Z70
AF3B 2014AF 2274 JSR SAF14 ;check for pointer < $A000 or > $E342
AF3E 901C 2275 BCC BAF5C ;yes, return
AF40 E054 2276 CPX 'T ;if first character of variable not T,
AF42 D018 2277 BNE BAF5C ;return
AF44 C0C9 2278 CPY $C9 ;if second character = "I$"
AF46 D014 2279 BNE BAF5C
AF48 2084AF 2280 JSR SAF84 ;get time in flp accu
AF4B 845E 2281 STY Z5E ;clear exponent base 10
AF4D 88 2282 DEY
AF4E 8471 2283 STY Z71 ;initialize output index
AF50 A006 2284 LDY $06 ;and # of digits before decimal point
AF52 845D 2285 STY Z5D
AF54 A024 2286 LDY Z24 ;set table index for time conversion
AF56 2068BE 2287 JSR SBE68 ;convert flp to string
AF59 4C6FB4 2288 JMP JB46F ;get description of string into flp accu
2289 ;
AF5C 60 2290 BAF5C RTS
2291 ;
AF5D 240E 2292 BAF5D BIT Z0E ;if integer variable
AF5F 100D 2293 BPL BAF6E
AF61 A000 2294 LDY $00
AF63 B164 2295 LDA (Z64),Y ;get integer value in AY
AF65 AA 2296 TAX
AF66 C8 2297 INY
AF67 B164 2298 LDA (Z64),Y
AF69 A8 2299 TAY
AF6A 8A 2300 TXA
AF6B 4C91B3 2301 JMP JB391 ;convert integer to flp
2302 ;
AF6E 2014AF 2303 BAF6E JSR SAF14 ;if not normal variable
AF71 902D 2304 BCC BAF40
AF73 E054 2305 CPX 'T ;and if first character is T
AF75 D01B 2306 BNE BAF92
AF77 C049 2307 CPY 'I ;and second character is I
AF79 D025 2308 BNE BAF40
AF7B 2084AF 2309 JSR SAF84 ;get time into flp accu
AF7E 98 2310 TYA
AF7F A2A0 2311 LDX $A0 ;set exponent
AF81 4C4FBC 2312 JMP JBC4F ;sign and guard bits

```

```

2314 ;get time in flp accu
2315 ;
AF84 20DEFF 2316 SAF84 JSR XFFDE ;fetch time
AF87 8664 2317 STX Z64
AF89 8463 2318 STY Z63 ;store time
AF8B 8565 2319 STA Z65
AF8D A000 2320 LDY $00
AF8F 8462 2321 STY Z62
AF91 60 2322 RTS
2323 ;
2324 ;continue to get value of variable
2325 ;
AF92 E053 2326 BAF92 CPX 'S ;if first character is S
AF94 D00A 2327 BNE BAF90
AF96 C054 2328 CPY 'T ;and second character is T
AF98 D006 2329 BNE BAF90
AF9A 20B7FF 2330 JSR XFFB7 ;get status word
AF9D 4C3CBC 2331 JMP JBC3C ;move signed # from A into flp accu
2332 ;
AFA0 A564 2333 BAF90 LDA Z64 ;for normal variables, put pointer in AY
AFA2 A465 2334 LDY Z65
AFA4 4CA2BB 2335 JMP SBBA2 ;load flp accu from AY
2336 ;
2337 ;apply function
2338 ;
AFA7 0A 2339 JAF7 ASL A ;index byte * 2
AFA8 48 2340 PHA ;save index
AFA9 AA 2341 TAX
AFAA 207300 2342 JSR X0073 ;get next character
AFAD E08F 2343 CPX $8F ;if function with more than 1 parameter
AFAF 9020 2344 BCC BAFD1
AFB1 20FAAE 2345 JSR SAEFA ;next character must be "("
AFB4 209EAD 2346 JSR SAD9E ;evaluate expression
AFB7 20FDAE 2347 JSR SAEFD ;next character must be ","
AFBA 208FAD 2348 JSR SAD8F ;result must be a string value
AFBD 68 2349 PLA ;restore index
AFBE AA 2350 TAX
AFBF A565 2351 LDA Z65 ;save pointer to description on stack
AFC1 48 2352 PHA
AFC2 A564 2353 LDA Z64
AFC4 48 2354 PHA
AFC5 8A 2355 TXA ;save index again
AFC6 48 2356 PHA
AFC7 209EB7 2357 JSR SB79E ;get next integer value into X
AFCA 68 2358 PLA ;restore index
AFCB A8 2359 TAY ;index in Y
AFCC 8A 2360 TXA
AFCD 48 2361 PHA ;save integer parameter
AFCE 4CD6AF 2362 JMP JAFD6
2363 ;
AFD1 20F1AE 2364 BAFD1 JSR BAEF1 ;get value in parenthesis
AFD4 68 2365 PLA ;restore index
AFD5 A8 2366 TAY
AFD6 B9EA9F 2367 JAFD6 LDA X9FEA,Y ;get pointer to function routine
AFD9 8555 2368 STA Z55 ;into Z55/Z56
AFDB B9EB9F 2369 LDA X9FEA+1,Y
AFDE 8556 2370 STA Z56
AFE0 205400 2371 JSR X0054 ;perform function
AFE3 4C8DAD 2372 JMP SAD8D ;return with correct string flag

```

		2374	;"OR" command	
		2375	;	
AFF6	A0FF	2376	WAFE6 LDY \$FF	;set parameter to complement operands
AFF8	2C	2377	.BY \$2C	;skip next instruction
		2378	;	
		2379	;"AND" command	
		2380	;	
AFF9	A000	2381	WAFE9 LDY \$00	;set no complement of operands
AFFB	840B	2382	STY Z0B	;store the parameter
AFFD	20BFB1	2383	JSR SB1BF	;convert left operand to integer
AFF0	A564	2384	LDA Z64	;get high byte
AFF2	450B	2385	EOR Z0B	;complement if OR
AFF4	8507	2386	STA Z07	;save result
AFF6	A565	2387	LDA Z65	;get low byte
AFF8	450B	2388	EOR Z0B	;complement if OR
AFFA	8508	2389	STA Z08	;save result
AFFC	20FCBB	2390	JSR SB1BF	;convert left operand to integer
AFFF	20BFB1	2391	JSR SB1BF	;convert right operand to integer
B002	A565	2392	LDA Z65	;get low byte
B004	450B	2393	EOR Z0B	;complement if OR
B006	2508	2394	AND Z08	;AND with left operand
B008	450B	2395	EOR Z0B	;complement if OR
B00A	A8	2396	TAY	;save result
B00B	A564	2397	LDA Z64	;get high byte
B00D	450B	2398	EOR Z0B	;complement if OR
B00F	2507	2399	AND Z07	;AND with other operand
B011	450B	2400	EOR Z0B	;complement if OR
B013	4C91B3	2401	JMP JB391	;convert integer from AY to flp accu

```

2403 ; "<=>" command
2404 ;
B016 2090AD 2405 WB016 JSR SAD90 ;check value of A
B019 B013 2406 BCS BB02E ;if string, go compare strings
B01B A56E 2407 LDA Z6E ;include sign bit
B01D 097F 2408 ORA $7F
B01F 256A 2409 AND Z6A ;into byte 1
B021 856A 2410 STA Z6A ;of fraction of second fip accu
B023 A969 2411 LDA <Z69
B025 A000 2412 LDY >Z69 ;set AY to second fip accu
B027 205BBC 2413 JSR SBC5B ;compare AY with fip accu
B02A AA 2414 TAX ;save result of comparison
B02B 4C61B0 2415 JMP JB061 ;check with <=> code
2416 ;
B02E A900 2417 BB02E LDA $00
B030 850D 2418 STA Z0D ;reset string flag
B032 C64D 2419 DEC Z4D ;also remove saved string flag
B034 20A6B6 2420 JSR SB6A6 ;de-allocate temp. string (right op)
B037 8561 2421 STA Z61
B039 8662 2422 STX Z62
B03B 8463 2423 STY Z63
B03D A56C 2424 LDA Z6C ;get text pointer of left operand
B03F A46D 2425 LDY Z6D
B041 20AAB6 2426 JSR SB6AA ;de-allocate temp. string (left op)
B044 866C 2427 STX Z6C ;save text pointer
B046 846D 2428 STY Z6D
B048 AA 2429 TAX ;save length of left operand
B049 38 2430 SEC
B04A E561 2431 SBC Z61 ;compare lengths (left-right)
B04C F008 2432 BEQ BB056 ;if equal, A=0
B04E A901 2433 LDA $01 ;set A to 1
B050 9004 2434 BCC BB056 ;if left operand longer,
B052 A661 2435 LDX Z61 ;get length of shorter string
B054 A9FF 2436 LDA $FF ;and set A to -1
B056 8566 2437 BB056 STA Z66 ;save sign of left-right
B058 A0FF 2438 LDY $FF ;set start index into strings
B05A E8 2439 INX
B05B C8 2440 BB05B INY ;point to next character
B05C CA 2441 DEX
B05D D007 2442 BNE BB066 ;if more characters, compare
B05F A666 2443 LDX Z66 ;if not, restore sign of left-right
B061 300F 2444 JB061 BMI BB072
B063 18 2445 CLC ;set correct C bit
B064 900C 2446 BCC BB072 ;JMP
B066 B16C 2447 BB066 LDA (Z6C),Y ;get character from left string
B068 D162 2448 CMP (Z62),Y ;compare to character from right string
B06A F0EF 2449 BEQ BB05B ;if equal, repeat
B06C A2FF 2450 LDX $FF ;X = -1 if character from left higher
B06E B002 2451 BCS BB072
B070 A201 2452 LDX $01 ;X = 1 if character from left lower
B072 E8 2453 BB072 INX ;X = 0, 1 or 2 for left > = <
B073 8A 2454 TXA
B074 2A 2455 ROL A ;convert to 1, 2 or 4
B075 2512 2456 AND Z12 ;mask with <=> code
B077 F002 2457 BEQ BB07B ;if zero, result = false (0)
B079 A9FF 2458 LDA $FF ;if non-zero, result = true (-1)
B07B 4C3CBC 2459 BB07B JMP JBC3C ;go get signed # from A into fip accu

```

```

2461 ;"DIM" command
2462 ;
BO7E 20FDAE 2463 BB07E JSR SAEFD ;SYNTAX Error if not ","
BO81 AA 2464 WB081 TAX
BO82 2090B0 2465 JSR SB090 ;gather name and pointer to variable
BO85 207900 2466 JSR X0079 ;get current character
BO88 DOF4 2467 BNE BB07E ;if not end of statement, repeat
BO8A 60 2468 RTS
2469 ;
2470 ;get name and pointer to a variable
2471 ;
BO8B A200 2472 SB08B LDX $00 ;set code for ref, not declaration
BO8D 207900 2473 JSR X0079 ;get current character
BO90 860C 2474 SB090 STX Z0C ;set reference/declaration flag
BO92 8545 2475 SB092 STA Z45 ;store first character of name
BO94 207900 2476 JSR X0079 ;get current character
BO97 2013B1 2477 JSR SB113 ;if not alphabetic
BO9A B003 2478 BCS BB09F
BO9C 4C08AF 2479 BB09C JMP JAF08 ;then SYNTAX Error message
2480 ;
BO9F A200 2481 BB09F LDX $00 ;set flags
BOA1 860D 2482 STX Z0D ;to not string
BOA3 860E 2483 STX Z0E ;and not integer
BOA5 207300 2484 JSR X0073 ;get next character
BOA8 9005 2485 BCC BBOAF ;if numeric
BOAA 2013B1 2486 JSR SB113
BOAD 9008 2487 BCC BBOBA ;or alphabetic
BOAF AA 2488 BBOAF TAX ;save second character in X
BOB0 207300 2489 BB0B0 JSR X0073 ;get next character
BOB3 90FB 2490 BCC BB0B0 ;until first non-numeric character
BOB5 2013B1 2491 JSR SB113
BOB8 B0F6 2492 BCS BB0B0 ;not alphabetic
BOBA C924 2493 BB0BA CMP "$" ;if terminator is "$"
BOBC D006 2494 BNE BBOC4
BOBE A9FF 2495 LDA $FF
BOC0 850D 2496 STA Z0D ;then set flag for strings
BOC2 D010 2497 BNE BBOD4
BOC4 C925 2498 BBOC4 CMP "Z" ;if terminator is "Z"
BOC6 D013 2499 BNE BBODB
BOC8 A510 2500 LDA Z10 ;and integers not allowed
BOCA D0D0 2501 BNE BB09C ;then SYNTAX Error
BOCC A980 2502 LDA $80
BOCE 850E 2503 STA Z0E ;else set integer flag and
BOD0 0545 2504 ORA Z45 ;add bit 7 to first char of name
BOD2 8545 2505 STA Z45
BOD4 8A 2506 BBOD4 TXA ;add bit 7 to 2nd char of name
BOD5 0980 2507 ORA $80 ;for both string and integer
BOD7 AA 2508 TAX
BOD8 207300 2509 JSR X0073 ;get next character
BODB 8646 2510 BBODB STX Z46 ;save second character name
BODD 38 2511 SEC
BODE 0510 2512 ORA Z10 ;if arrays allowed
BOE0 E928 2513 SBC "(" ;and next character is "("
BOE2 D003 2514 BNE BBOE7
BOE4 4CD1B1 2515 JMP JB1D1 ;handle dimensioned variable
2516 ;
BOE7 A000 2517 BBOE7 LDY $00
BOE9 8410 2518 STY Z10 ;reset no integers or arrays switch
BOEB A52D 2519 LDA Z2D
BOED A62E 2520 LDX Z2E ;AX=pointer to variable name table

```

```

BOEF 8660    2521 BBOEF STX Z60
BOF1 855F    2522 BBOF1 STA Z5F
BOF3 E430    2523     CPX Z30      ;if at end of name table
BOF5 D004    2524     BNE BBOFB
BOF7 C52F    2525     CMP Z2F
BOF9 FO22    2526     BEQ BB11D    ;then variable not found
BOFB A545    2527 BBOFB LDA Z45
BOFD D15F    2528     CMP (Z5F),Y  ;if name matches
BOFF D008    2529     BNE BB109
B101 A546    2530     LDA Z46
B103 C8      2531     INY
B104 D15F    2532     CMP (Z5F),Y
B106 F07D    2533     BEQ BB185    ;then variable found
B108 88      2534     DEY
B109 18      2535 BB109 CLC
B10A A55F    2536     LDA Z5F      ;set AX to entry + 7
B10C 6907    2537     ADC $07
B10E 90E1    2538     BCC BBOF1
B110 E8      2539     INX
B111 D0DC    2540     BNE BBOEF    ;continue search
          2541 ;
          2542 ;check character in A
          2543 ;
          2544 ;return with C=1 if alphabetic
          2545 ;return with C=0 if not
          2546 ;
B113 C941    2547 SB113 CMP 'A      ;if character < "A"
B115 9005    2548     BCC BB11C    ;return with C = 0
B117 E95B    2549     SBC 'Z+1    ;if character > "Z" then C = 0
B119 38      2550     SEC
B11A E9A5    2551     SBC $A5      ;else return with C=1
B11C 60      2552 BB11C RTS
          2553 ;
          2554 ;variable not found
          2555 ;
B11D 68      2556 BB11D PLA
B11E 48      2557     PHA      ;if low byte of return address = $2A
B11F C92A    2558     CMP $2A    (if called from GET VALUE OF VARIABLE)
B121 D005    2559     BNE BB128
B123 A913    2560 BB123 LDA $13    ;return with AY pointing to
B125 A0BF    2561     LDY $BF    ;a dummy variable (value 0)
B127 60      2562     RTS
          2563 ;
B128 A545    2564 BB128 LDA Z45
B12A A446    2565     LDY Z46
B12C C954    2566     CMP 'T      ;if name = TI$
B12E D00B    2567     BNE BB13B
B130 C0C9    2568     CPY $C9
B132 F0EF    2569     BEQ BB123    ;return with dummy value
B134 C049    2570     CPY 'I      ;if name=TI
B136 D003    2571     BNE BB13B
B138 4C08AF  2572 BB138 JMP JAF08  ;then SYNTAX Error
          2573 ;
B13B C953    2574 BB13B CMP 'S      ;if name=ST
B13D D004    2575     BNE BB143
B13F C054    2576     CPY 'T
B141 F0F5    2577     BEQ BB138    ;then SYNTAX Error
B143 A52F    2578 BB143 LDA Z2F    ;set AY to pointer to end of name table
B145 A430    2579     LDY Z30
B147 855F    2580     STA Z5F    ;and save as start of input for move

```


B149	8460	2581	STY Z60	
B14B	A531	2582	LDA Z31	;AY=pointer to end of array area
B14D	A432	2583	LDY Z32	
B14F	855A	2584	STA Z5A	
B151	845B	2585	STY Z5B	
B153	18	2586	CLC	
B154	6907	2587	ADC \$07	;add 7 to pointer for one entry
B156	9001	2588	BCC BB159	
B158	C8	2589	INY	
B159	8558	2590	BB159 STA Z58	;into pointer to end of output for move
B15B	8459	2591	STY Z59	
B15D	20B8A3	2592	JSR SA3B8	;check for space and move bytes
B160	A558	2593	LDA Z58	
B162	A459	2594	LDY Z59	;AY=beginning of output area
B164	C8	2595	INY	
B165	852F	2596	STA Z2F	;for move routine
B167	8430	2597	STY Z30	;move to pointer to end of name table
B169	A000	2598	LDY \$00	
B16B	A545	2599	LDA Z45	
B16D	915F	2600	STA (Z5F),Y	;store first byte of name in byte 0
B16F	C8	2601	INY	
B170	A546	2602	LDA Z46	
B172	915F	2603	STA (Z5F),Y	;store second byte of name in byte 1
B174	A900	2604	LDA \$00	
B176	C8	2605	INY	
B177	915F	2606	STA (Z5F),Y	;clear bytes 2-6 (value)
B179	C8	2607	INY	
B17A	915F	2608	STA (Z5F),Y	
B17C	C8	2609	INY	
B17D	915F	2610	STA (Z5F),Y	
B17F	C8	2611	INY	
B180	915F	2612	STA (Z5F),Y	
B182	C8	2613	INY	
B183	915F	2614	STA (Z5F),Y	
		2615	;	
		2616	;variable found	
		2617	;	
B185	A55F	2618	BB185 LDA Z5F	
B187	18	2619	CLC	;return pointer to entry
B188	6902	2620	ADC \$02	;plus 2
B18A	A460	2621	LDY Z60	
B18C	9001	2622	BCC BB18F	
B18E	C8	2623	INY	
B18F	8547	2624	BB18F STA Z47	;in variable address and AY
B191	8448	2625	STY Z48	
B193	60	2626	RTS	

```

        2628 ;compute pointer to array body
        2629 ;
B194 A50B 2630 SB194 LDA Z0B      ;get # of dimensions
B196 0A   2631     ASL A          ;* 2
B197 6905 2632     ADC $05        ;add fixed overhead
B199 655F 2633     ADC Z5F        ;add low byte of array pointer
B19B A460 2634     LDY Z60        ;get high byte of array pointer
B19D 9001 2635     BCC B1A0
B19F C8   2636     INY            ;increment if carry
B1A0 8558 2637 BB1A0 STA Z58      ;save pointer to body of array
B1A2 8459 2638     STY Z59
B1A4 60   2639     RTS
        2640 ;
        2641 ;flp number for conversion to integer
        2642 ;
B1A5 908000 2643 TB1A5 .BY $90,$80,$00,$00,$00
        2644 ;
        2645 ;routine to convert floating point number to fixed point
        2646 ;
B1AA 20BF81 2647     JSR SB1BF      ;convert flp to fixed point
B1AD A564 2648     LDA Z64
B1AF A465 2649     LDY Z65        ;return with value in AY
B1B1 60   2650     RTS
        2651 ;
        2652 ;convert value from statement into integer
        2653 ;
B1B2 207300 2654 SB1B2 JSR X0073   ;get next character
B1B5 209EAD 2655     JSR SAD9E      ;get value from statement
B1B8 208DAD 2656 SB1B8 JSR SAD8D   ;check for non-string
B1BB A566 2657     LDA Z66        ;if flp accu negative,
B1BD 300D 2658     BMI B11CC       ;ILLEGAL QUANTITY Error
        2659 ;
        2660 ;convert flp number to integer
        2661 ;
B1BF A561 2662 SB1BF LDA Z61      ;get exponent
B1C1 C990 2663     CMP $90        ;if < 16
B1C3 9009 2664     BCC B11CE      ;convert
B1C5 A9A5 2665     LDA <TB1A5     ;let AY point to flp value -32768
B1C7 A0B1 2666     LDY >TB1A5
B1C9 205BBC 2667     JSR SBC5B     ;compare AY to flp accu
B1CC D07A 2668 BB1CC BNE JB248    ;if not >, ILLEGAL QUANTITY Error
B1CE 4C9BBC 2669 BB1CE JMP SBC9B  ;convert flp accu to 4 byte integer

```

		2671	;get pointer to dimensioned variable
		2672	;
B1D1 A50C	2673	JB1D1 LDA Z0C	;save reference flag
B1D3 050E	2674	ORA Z0E	;and integer flag
B1D5 48	2675	PHA	
B1D6 A50D	2676	LDA Z0D	;save string flag
B1D8 48	2677	PHA	
B1D9 A000	2678	LDY \$00	;initialize count # of dimensions
B1DB 98	2679	BB1DB TYA	
B1DC 48	2680	PHA	;save count
B1DD A546	2681	LDA Z46	
B1DF 48	2682	PHA	;save second character of name
B1E0 A545	2683	LDA Z45	
B1E2 48	2684	PHA	;save first character of name
B1E3 20B2B1	2685	JSR SB1B2	;convert value from statement to integer
B1E6 68	2686	PLA	
B1E7 8545	2687	STA Z45	;restore first character of name
B1E9 68	2688	PLA	
B1EA 8546	2689	STA Z46	;restore second character of name
B1EC 68	2690	PLA	
B1ED A8	2691	TAY	;restore count
B1EE BA	2692	TSX	;get stack pointer
B1EF B0201	2693	LDA X0102,X	
B1F2 48	2694	PHA	;move saved flags to top of stack
B1F3 B0101	2695	LDA X0101,X	
B1F6 48	2696	PHA	
B1F7 A564	2697	LDA Z64	;save array index in their place
B1F9 9D0201	2698	STA X0102,X	
B1FC A565	2699	LDA Z65	
B1FE 9D0101	2700	STA X0101,X	
B201 C8	2701	INY	;increment dimension pointer
B202 207900	2702	JSR X0079	;get current character
B205 C92C	2703	CMP "	;if ","
B207 F0D2	2704	BEQ BB1DB	;repeat for next array index
B209 840B	2705	STY Z0B	;save total number of dimensions
B20B 20F7AE	2706	JSR SAEF7	;skip ")"
B20E 68	2707	PLA	
B20F 850D	2708	STA Z0D	;restore string flag
B211 68	2709	PLA	
B212 850E	2710	STA Z0E	;restore integer flag
B214 297F	2711	AND \$7F	
B216 850C	2712	STA Z0C	;and reference flag
B218 A62F	2713	LDX Z2F	
B21A A530	2714	LDA Z30	;AX = pointer to array area
B21C 865F	2715	BB21C STX Z5F	
B21E 8560	2716	STA Z60	;save in temporary pointer
B220 C532	2717	CMP Z32	
B222 D004	2718	BNE BB228	
B224 E431	2719	CPX Z31	;if end of array area reached
B226 F039	2720	BEQ BB261	;then array not found
B228 A000	2721	BB228 LDY \$00	
B22A B15F	2722	LDA (Z5F),Y	;get first character of name
B22C C8	2723	INY	
B22D C545	2724	CMP Z45	;if equal
B22F D006	2725	BNE BB237	
B231 A546	2726	LDA Z46	
B233 D15F	2727	CMP (Z5F),Y	;and second character of name also
B235 F016	2728	BEQ BB24D	;then array found
B237 C8	2729	BB237 INY	
B238 B15F	2730	LDA (Z5F),Y	;get low byte of length

B23A 18	2731	CLC	
B23B 655F	2732	ADC Z5F	;add to array pointer
B23D AA	2733	TAX	;and save result in X
B23E C8	2734	INY	
B23F B15F	2735	LDA (Z5F),Y	;get high byte of length
B241 6560	2736	ADC Z60	;add to array pointer
B243 90D7	2737	BCC BB21C	;repeat if not overflow
B245 A212	2738 BB245	LDX \$12	;point to BAD SUBSCRIPT Error message
B247 2C	2739	.BY \$2C	;skip next instruction
	2740 ;		
B248 A20E	2741 JB248	LDX \$0E	;point to ILLEGAL QUANTITY Error message
B24A 4C37A4	2742 BB24A	JMP JA437	;print message
B24D A213	2743 BB24D	LDX \$13	;point to BAD SUBSCRIPT Error message
B24F A50C	2744	LDA Z0C	;if declaration
B251 D0F7	2745	BNE BB24A	;then fatal error
B253 2094B1	2746	JSR SB194	;compute pointer to array body
B256 A50B	2747	LDA Z0B	;if actual # of dimensions
B258 A004	2748	LDY \$04	
B25A D15F	2749	CMP (Z5F),Y	;not equal to declared #
B25C D0E7	2750	BNE BB245	;then error
B25E 4CEAB2	2751	JMP JB2EA	;else compute array reference

	2753	allocate array	
	2754	;	
B261 2094B1	2755	BB261 JSR SB194	;compute pointer to virtual array body
B264 2008A4	2756	JSR SA408	;check array area for overflow
B267 A000	2757	LDY \$00	
B269 8472	2758	STY Z72	;initialize high byte of length
B26B A205	2759	LDX \$05	;set length per element
B26D A545	2760	LDA Z45	;get first character of name
B26F 915F	2761	STA (Z5F),Y	;into + 0 of array header
B271 1001	2762	BFL BB274	;if integer
B273 CA	2763	DEX	;adjust length per element
B274 C8	2764	BB274 INY	
B275 A546	2765	LDA Z46	;move second character of name
B277 915F	2766	STA (Z5F),Y	;into + 1 of array header
B279 1002	2767	BFL BB27D	;if integer or string
B27B CA	2768	DEX	
B27C CA	2769	DEX	;correct length per element
B27D 8671	2770	BB27D STX Z71	;set initial low byte of length
B27F A50B	2771	LDA Z0B	;move # of dimensions
B281 C8	2772	INY	
B282 C8	2773	INY	
B283 C8	2774	INY	
B284 915F	2775	STA (Z5F),Y	;into + 4 of array header
B286 A20B	2776	BB286 LDX \$0B	;set default dimension (11)
B288 A900	2777	LDA \$00	
B28A 240C	2778	BIT Z0C	;if declaration
B28C 5008	2779	BVC BB296	;skip
B28E 68	2780	PLA	;else get limit from stack
B28F 18	2781	CLC	
B290 6901	2782	ADC \$01	;add 1 for element # 0
B292 AA	2783	TAX	
B293 68	2784	PLA	;limit into AX
B294 6900	2785	ADC \$00	
B296 C8	2786	BB296 INY	
B297 915F	2787	STA (Z5F),Y	;move limit into next byte of header
B299 C8	2788	INY	
B29A 8A	2789	TXA	
B29B 915F	2790	STA (Z5F),Y	;low byte also
B29D 204CB3	2791	JSR SB34C	;AX = length * limit
B2A0 8671	2792	STX Z71	
B2A2 8572	2793	STA Z72	;save into length
B2A4 A422	2794	LDY Z22	;restore index into array header
B2A6 C60B	2795	DEC Z0B	;decrement # of dimensions
B2A8 D0DC	2796	BNE BB286	;repeat until done
B2AA 6559	2797	ADC Z59	;add high byte of length to body pointer
B2AC B05D	2798	BCS BB30B	;if overflow, OUT OF MEMORY Error
B2AE 8559	2799	STA Z59	;save pointer to end of body
B2B0 A8	2800	TAY	;also in Y
B2B1 8A	2801	TXA	
B2B2 6558	2802	ADC Z58	;add low byte of length to body pointer
B2B4 9003	2803	BCC BB2B9	;if overflow
B2B6 C8	2804	INY	;increment high byte
B2B7 F052	2805	BEQ BB30B	;if overflow, OUT OF MEMORY Error
B2B9 2008A4	2806	BB2B9 JSR SA408	;check array area for overflow
B2BC 8531	2807	STA Z31	
B2BE 8432	2808	STY Z32	;set new end of array area
B2C0 A900	2809	LDA \$00	
B2C2 E672	2810	INC Z72	;get length
B2C4 A471	2811	LDY Z71	
B2C6 F005	2812	BEQ BB2CD	;if low byte of length not zero

B2C8 88	2813	BB2C8 DEY	;decrement index
B2C9 9158	2814	STA (Z58),Y	;store zero in array byte
B2CB D0FB	2815	BNE BB2C8	;until low byte of length is zero
B2CD C659	2816	BB2CD DEC Z59	;then set pointer to next block of 256
B2CF C672	2817	DEC Z72	;and decrement high byte of length
B2D1 D0F5	2818	BNE BB2C8	;repeat until length is zero
B2D3 E659	2819	INC Z59	
B2D5 38	2820	SEC	
B2D6 A531	2821	LDA Z31	;get address of end of array area
B2D8 E55F	2822	SBC Z5F	;minus pointer to array area
B2DA A002	2823	LDY \$02	
B2DC 915F	2824	STA (Z5F),Y	;store low byte in + 2 of array header
B2DE A532	2825	LDA Z32	;move high byte of difference
B2E0 C8	2826	INY	
B2E1 E560	2827	SBC Z60	
B2E3 915F	2828	STA (Z5F),Y	;into + 3 of array header
B2E5 A50C	2829	LDA Z0C	;if declaration
B2E7 D062	2830	BNE BB34B	;then return
B2E9 C8	2831	INY	;else point to declared # of dimensions
	2832	;	
	2833	;compute reference to array element	
	2834	;	
B2EA B15F	2835	JB2EA LDA (Z5F),Y	;get # of dimensions from array header
B2EC 850B	2836	STA Z0B	;save it
B2EE A900	2837	LDA \$00	
B2F0 8571	2838	STA Z71	;initialize offset in body
B2F2 8572	2839	BB2F2 STA Z72	
B2F4 C8	2840	INY	
B2F5 68	2841	PLA	;get low byte of index from stack
B2F6 AA	2842	TAX	;into X
B2F7 8564	2843	STA Z64	;and flp accu
B2F9 68	2844	PLA	;get high byte of index from stack
B2FA 8565	2845	STA Z65	;into A and flp accu
B2FC D15F	2846	CMP (Z5F),Y	;compare index to limit
B2FE 900E	2847	BCC BB30E	;if lower, OK
B300 D006	2848	BNE BB308	;if higher, BAD SUBSCRIPT Error
B302 C8	2849	INY	;if equal
B303 8A	2850	TXA	
B304 D15F	2851	CMP (Z5F),Y	;compare low bytes
B306 9007	2852	BCC BB30F	;if lower, OK
B308 4C45B2	2853	BB308 JMP BB245	;else BAD SUBSCRIPT Error
B30B 4C35A4	2854	BB30B JMP BA435	;OUT OF MEMORY Error
	2855	;	
B30E C8	2856	BB30E INY	
B30F A572	2857	BB30F LDA Z72	;if total index so far = 0
B311 0571	2858	ORA Z71	
B313 18	2859	CLC	
B314 F00A	2860	BEQ BB320	;skip multiplication
B316 204CB3	2861	JSR BB34C	;XY = offset * limit
B319 8A	2862	TXA	
B31A 6564	2863	ADC Z64	;add low byte of array index to A
B31C AA	2864	TAX	
B31D 98	2865	TYA	
B31E A422	2866	LDY Z22	;restore index to array header
B320 6565	2867	BB320 ADC Z65	;add high byte of array index to A
B322 8671	2868	STX Z71	;save low byte of new offset
B324 C60B	2869	DEC Z0B	;if not last index,
B326 D0CA	2870	BNE BB2F2	;repeat for next index
B328 8572	2871	STA Z72	;save high byte of offset
B32A A205	2872	LDX \$05	;set length per element for flp #

B32C	A545	2873	LDA Z45	
B32E	1001	2874	BPL BB331	;if integer
B330	CA	2875	DEX	;correct length per element
B331	A546	2876	BB331 LDA Z46	
B333	1002	2877	BPL BB337	;if integer or string
B335	CA	2878	DEX	
B336	CA	2879	DEX	;correct length per element
B337	8628	2880	BB337 STX Z28	
B339	A900	2881	LDA \$00	
B33B	2055B3	2882	JSR SB355	;XY = offset * length per element
B33E	8A	2883	TXA	
B33F	6558	2884	ADC Z58	;add to pointer to array body
B341	8547	2885	STA Z47	;store result into variable address
B343	98	2886	TYA	
B344	6559	2887	ADC Z59	
B346	8548	2888	STA Z48	;high byte also
B348	AB	2889	TAY	
B349	A547	2890	LDA Z47	;return with AY = variable address
B34B	60	2891	BB34B RTS	

```

2893 ;XY = XA = length * limit from array data
2894 ;
B34C 8422 2895 SB34C STY Z22 ;save index in array header
B34E B15F 2896 LDA (Z5F),Y ;get low byte of limit
B350 8528 2897 STA Z28 ;save it
B352 88 2898 DEY
B353 B15F 2899 LDA (Z5F),Y ;get high byte of limit
B355 8529 2900 SB355 STA Z29 ;and save it too
B357 A910 2901 LDA $10 ;set # bits for multiplication
B359 855D 2902 STA Z5D
B35B A200 2903 LDX $00 ;initialize result fields
B35D A000 2904 LDY $00
B35F 8A 2905 BB35F TXA
B360 0A 2906 ASL A ;XY * 2
B361 AA 2907 TAX
B362 98 2908 TYA
B363 2A 2909 ROL A
B364 AB 2910 TAY
B365 B0A4 2911 BCS BB30B ;if overflow, OUT OF MEMORY Error
B367 0671 2912 ASL Z71
B369 2672 2913 ROL Z72 ;shift high bit out of length
B36B 900B 2914 BCC BB378
B36D 18 2915 CLC ;if set
B36E 8A 2916 TXA
B36F 6528 2917 ADC Z28 ;add low byte of limit to XY
B371 AA 2918 TAX
B372 98 2919 TYA
B373 6529 2920 ADC Z29 ;high byte also
B375 AB 2921 TAY
B376 B093 2922 BCS BB30B ;if overflow, OUT OF MEMORY Error
B378 C65D 2923 BB378 DEC Z5D
B37A DOE3 2924 BNE BB35F ;repeat for all bits of length
B37C 60 2925 RTS

```


	2927	;"FRE" command	
	2928	;	
B37D A50D	2929	WB37D LDA Z0D	;if string flag set
B37F F003	2930	BEQ BB384	
B381 20A6B6	2931	JSR SB6A6	;de-allocate temporary string storage
B384 2026B5	2932	BB384 JSR SB526	;perform garbage clean-up
B387 38	2933	SEC	
B388 A533	2934	LDA Z33	;compute difference between
B38A E531	2935	SBC Z31	;pointer to allocated string area
B38C A8	2936	TAY	;and pointer to end of array area
B38D A534	2937	LDA Z34	
B38F E532	2938	SBC Z32	;convert length into flp accu
	2939	;	
	2940	;routine to convert integer to floating point	
	2941	;	
B391 A200	2942	JB391 LDX \$00	
B393 860D	2943	STX Z0D	;clear string flag
B395 8562	2944	STA Z62	;store integer from AY into flp accu
B397 8463	2945	STY Z63	
B399 A290	2946	LDX \$90	;load exponent (16)
B39B 4C44BC	2947	JMP JBC44	;convert to floating point #
	2948	;	
	2949	;"POS" command	
	2950	;	
B39E 38	2951	WB39E SEC	;set carry to read cursor position
B39F 20F0FF	2952	JSR XFFF0	;perform read cursor position into XY
B3A2 A900	2953	JB3A2 LDA \$00	
B3A4 FOEB	2954	BEQ JB391	;convert integer into flp accu
	2955	;	
	2956	;check for non-direct mode	
	2957	;	
B3A6 A63A	2958	SB3A6 LDX Z3A	;get program/direct flag
B3A8 E8	2959	INX	
B3A9 DOA0	2960	BNE BB34B	;if direct,
B3AB A215	2961	LDX \$15	;point to ILLEGAL DIRECT Error message
B3AD 2C	2962	.BY \$2C	;skip next instruction
B3AE A21B	2963	BB3AE LDX \$1B	;point to UNDEF'D FUNCTION Error message
B3B0 4C37A4	2964	JMP JA437	;print error

```

                2966 ;"DEF" command
                2967 ;
B3B3 20E1B3 2968 WB3B3 JSR SB3E1 ;get function name
B3B6 20A6B3 2969 JSR SB3A6 ;check for non-direct mode
B3B9 20FAAE 2970 JSR SAEFA ;next character must be "("
B3BC A980 2971 LDA $80
B3BE 8510 2972 STA Z10 ;set no integers/array elements allowed
B3C0 208BB0 2973 JSR SB08B ;gather name and get pointer to variable
B3C3 208DAD 2974 JSR SAD8D ;check value to be non-string
B3C6 20F7AE 2975 JSR SAEF7 ;next value must be ")"
B3C9 A9B2 2976 LDA $E2
B3CB 20FFAE 2977 JSR SAEFF ;next character must be "-"
B3CE 48 2978 PHA ;save first character of definition
B3CF A548 2979 LDA Z48 ;save pointer to variable of function
B3D1 48 2980 PHA
B3D2 A547 2981 LDA Z47
B3D4 48 2982 PHA
B3D5 A57B 2983 LDA Z7B ;save current character pointer
B3D7 48 2984 PHA
B3D8 A57A 2985 LDA Z7A
B3DA 48 2986 PHA
B3DB 20F8A8 2987 JSR JA8F8 ;execute command DATA
B3DE 4C4FB4 2988 JMP JB44F ;set value for function name
                2989 ;
                2990 ;get function name
                2991 ;
B3E1 A9A5 2992 SB3E1 LDA $A5 ;if next character not code for FN
B3E3 20FFAE 2993 JSR SAEFF ;then SYNTAX Error
B3E6 0980 2994 ORA $80 ;else set bit 7 of
B3E8 8510 2995 STA Z10 ;no integers or array elements flag
B3EA 2092B0 2996 JSR SB092 ;gather name and get pointer to variable
B3ED 854E 2997 STA Z4E ;save pointer to variable
B3EF 844F 2998 STY Z4F
B3F1 4C8DAD 2999 JMP SAD8D ;check value to be non-string
                3000 ;
                3001 ;expand FN call
                3002 ;
B3F4 20E1B3 3003 JB3F4 JSR SB3E1 ;get function name
B3F7 A54F 3004 LDA Z4F
B3F9 48 3005 PHA ;save pointer to name on variable stack
B3FA A54E 3006 LDA Z4E
B3FC 48 3007 PHA
B3FD 20F1AE 3008 JSR BAEF1 ;get parathensized value
B400 208DAD 3009 JSR SAD8D ;check that value is non-string
B403 68 3010 PLA
B404 854E 3011 STA Z4E ;restore pointer to variable name
B406 68 3012 PLA
B407 854F 3013 STA Z4F
B409 A002 3014 LDY $02
B40B B14E 3015 LDA (Z4E),Y ;get index to variable
B40D 8547 3016 STA Z47 ;save pointer to formal parameters
B40F AA 3017 TAX
B410 C8 3018 INY
B411 B14E 3019 LDA (Z4E),Y ;high byte also
B413 F099 3020 BEQ BB3AE ;if data byte + 3 is zero, fatal error
B415 8548 3021 STA Z48
B417 C8 3022 INY
B418 B147 3023 BB418 LDA (Z47),Y ;save formal parameters on stack
B41A 48 3024 PHA
B41B 88 3025 DEY

```

B41C 10FA	3026	BPL BB418	
B41E A448	3027	LDY Z48	;get pointer to formal parameters in XY
B420 20D4BB	3028	JSR SBB4	;store flip accu into formal parameters
B423 A57B	3029	LDA Z7B	
B425 48	3030	PHA	;save current character pointer on stack
B426 A57A	3031	LDA Z7A	
B428 48	3032	PHA	
B429 B14E	3033	LDA (Z4E),Y	;move current pointer of definition
B42B 857A	3034	STA Z7A	;into current character pointer
B42D C8	3035	INY	
B42E B14E	3036	LDA (Z4E),Y	
B430 857B	3037	STA Z7B	
B432 A548	3038	LDA Z48	
B434 48	3039	PHA	
B435 A547	3040	LDA Z47	
B437 48	3041	PHA	
B438 208AAD	3042	JSR SAD8A	;get next non-string value
B43B 68	3043	PLA	;set pointer to formal parameters
B43C 854E	3044	STA Z4E	
B43E 68	3045	PLA	
B43F 854F	3046	STA Z4F	
B441 207900	3047	JSR X0079	;get current character
B444 F003	3048	BEQ BB449	;if not zero or ":"
B446 4C08AF	3049	JMP JAF08	;SYNTAX Error
	3050 ;		
B449 68	3051	BB449 PLA	
B44A 857A	3052	STA Z7A	;restore current character pointer
B44C 68	3053	PLA	
B44D 857B	3054	STA Z7B	;and restore formal parameters
B44F A000	3055	JB44F LDY \$00	;set index into variable
B451 68	3056	PLA	
B452 914E	3057	STA (Z4E),Y	;save stack into variable, byte + 0
B454 68	3058	PLA	
B455 C8	3059	INY	
B456 914E	3060	STA (Z4E),Y	;byte + 1
B458 68	3061	PLA	
B459 C8	3062	INY	
B45A 914E	3063	STA (Z4E),Y	;byte + 2
B45C 68	3064	PLA	
B45D C8	3065	INY	
B45E 914E	3066	STA (Z4E),Y	;byte + 3
B460 68	3067	PLA	
B461 C8	3068	INY	
B462 914E	3069	STA (Z4E),Y	;byte + 4
B464 60	3070	RTS	

	3072	;	"STR\$" command	
	3073	;		
B465	208DAD	3074	WB465 JSR SAD8D	;check if parameter non-string
B468	A000	3075	LDY \$00	;start output area at \$00FF
B46A	20DFBD	3076	JSR SBDDF	;convert to string
B46D	68	3077	PLA	;remove own return address
B46E	68	3078	PLA	
B46F	A9FF	3079	JB46F LDA \$FF	;set AY to point to string
B471	A000	3080	LDY \$00	;on bottom of stack
B473	F012	3081	BEQ SB487	;get description of constant string
B475	A664	3082	SB475 LDX Z64	;move pointer to string description
B477	A465	3083	LDY Z65	
B479	8650	3084	STX Z50	;into Z50/Z51
B47B	8451	3085	STY Z51	
		3086	;	
		3087	;allocate area according to A	
		3088	;	
B47D	20F4B4	3089	SB47D JSR SB4F4	;allocate area
B480	8662	3090	STX Z62	;store pointer
B482	8463	3091	STY Z63	
B484	8561	3092	STA Z61	;store length
B486	60	3093	RTS	

```

3095 ;get description of constant string into flp accu
3096 ;
B487 A222 3097 SB487 LDX ``
B489 8607 3098 STX Z07 ;set both terminators
B48B 8608 3099 STX Z08
B48D 856F 3100 SB48D STA Z6F ;store pointer to beginning of string
B48F 8470 3101 STY Z70 ;in temporary pointer
B491 8562 3102 STA Z62 ;and in flp accu
B493 8463 3103 STY Z63
B495 A0FF 3104 LDY $FF ;initialize string index
B497 C8 3105 BB497 INY
B498 B16F 3106 LDA (Z6F),Y ;get next character
B49A F00C 3107 BEQ BB4A8 ;if zero
B49C C507 3108 CMP Z07
B49E F004 3109 BEQ BB4A4 ;or first terminator,
B4A0 C508 3110 CMP Z08
B4A2 D0F3 3111 BNE BB497 ;or second terminator, then end found
B4A4 C922 3112 BB4A4 CMP `` ;if terminator = quote
B4A6 F001 3113 BEQ BB4A9
B4A8 18 3114 BB4A8 CLC
B4A9 8461 3115 BB4A9 STY Z61 ;save length
B4AB 98 3116 TYA
B4AC 656F 3117 ADC Z6F
B4AE 8571 3118 STA Z71 ;and set pointer beyond string
B4B0 A670 3119 LDX Z70
B4B2 9001 3120 BCC BB4B5
B4B4 E8 3121 INX
B4B5 8672 3122 BB4B5 STX Z72
B4B7 A570 3123 LDA Z70 ;if from statement in Direct mode,
B4B9 F004 3124 BEQ BB4BF
B4BB C902 3125 CMP $02
B4BD D00B 3126 BNE BB4CA
B4BF 98 3127 BB4BF TYA ;get length
B4C0 2075B4 3128 JSR SB475 ;allocate area
B4C3 A66F 3129 LDX Z6F
B4C5 A470 3130 LDY Z70 ;save pointer to it
B4C7 2088B6 3131 JSR SB688 ;move string into it

```

```

3133 ;save descriptor from Z61-Z63 to descriptor stack
3134 ;
B4CA A616 3135 BB4CA LDX Z16 ;get descriptor stack index
B4CC E022 3136 CPX <Z22 ;if at $22
B4CE D005 3137 BNE BB4D5
B4D0 A219 3138 LDX $19 ;point to FORMULA TOO COMPLEX Error
B4D2 4C37A4 3139 BB4D2 JMP JA437 ;print error
B4D5 A561 3140 BB4D5 LDA Z61 ;else move descriptor
B4D7 9500 3141 STA Z00,X ;to descriptor stack
B4D9 A562 3142 LDA Z62
B4DB 9501 3143 STA Z01,X
B4DD A563 3144 LDA Z63
B4DF 9502 3145 STA Z02,X
B4E1 A000 3146 LDY $00 ;set pointer to descriptor
B4E3 8664 3147 STX Z64 ;in flp accu
B4E5 8465 3148 STY Z65
B4E7 8470 3149 STY Z70 ;clear guard bit
B4E9 88 3150 DEY
B4EA 840D 3151 STY Z0D ;set string flag
B4EC 8617 3152 STX Z17 ;set previous descriptor stack index
B4EE E8 3153 INX
B4EF E8 3154 INX
B4F0 E8 3155 INX
B4F1 8616 3156 STX Z16 ;set new descriptor stack index
B4F3 60 3157 RTS

```

```

3159 ;allocate # of bytes in A
3160 ;
B4F4 460F 3161 SB4F4 LSR ZOF ;clear error flag
B4F6 48 3162 BB4F6 PHA ;save length
B4F7 49FF 3163 EOR $FF
B4F9 38 3164 SEC
B4FA 6533 3165 ADC Z33 ;subtract length from string storage ptr
B4FC A434 3166 LDY Z34
B4FE B001 3167 BCS BB501
B500 88 3168 DEY ;high byte also
B501 C432 3169 BB501 CPY Z32 ;if above
B503 9011 3170 BCC BB516 ;array area
B505 D004 3171 BNE BB50B
B507 C531 3172 CMP Z31
B509 900B 3173 BCC BB516
B50B 8533 3174 BB50B STA Z33 ;set new string storage pointer
B50D 8434 3175 STY Z34
B50F 8535 3176 STA Z35 ;set utility string pointer also
B511 8436 3177 STY Z36
B513 AA 3178 TAX ;and set AX
B514 68 3179 PLA ;restore length
B515 60 3180 RTS
3181 ;
B516 A210 3182 BB516 LDX $10
B518 A50F 3183 LDA ZOF ;if no memory left
B51A 30B6 3184 BMI BB4D2 ;print OUT OF MEMORY Error
B51C 2026B5 3185 JSR SB526 ;else perform garbage clean-up
B51F A980 3186 LDA $80
B521 850F 3187 STA ZOF ;clear flag
B523 68 3188 PLA ;restore # bytes to allocate
B524 D0D0 3189 BNE BB4F6 ;and perform allocation
3190 ;
3191 ;string garbage clean-up
3192 ;
B526 A637 3193 SB526 LDX Z37 ;get memory limit
B528 A538 3194 LDA Z38
B52A 8633 3195 JB52A STX Z33 ;set new string storage pointer
B52C 8534 3196 STA Z34
B52E A000 3197 LDY $00 ;set no descriptor of highest text yet
B530 844F 3198 STY Z4F
B532 844E 3199 STY Z4E
B534 A531 3200 LDA Z31 ;get end of array area
B536 A632 3201 LDX Z32
B538 855F 3202 STA Z5F ;initialize value for highest text ptr
B53A 8660 3203 STX Z60
B53C A919 3204 LDA <Z19 ;AX = ptr to bottom of descriptor stack
B53E A200 3205 LDX >Z19
B540 8522 3206 STA Z22 ;and store in temporary pointer
B542 8623 3207 STX Z23
B544 C516 3208 BB544 CMP Z16
B546 F005 3209 BEQ BB54D ;if not top index of descriptor stack
B548 20C7B5 3210 JSR SB5C7 ;find highest text area
B54B F0F7 3211 BEQ BB544
B54D A907 3212 BB54D LDA $07 ;set step to 7
B54F 8553 3213 STA Z53
B551 A52D 3214 LDA Z2D ;move pointer to name table
B553 A62E 3215 LDX Z2E
B555 8522 3216 STA Z22 ;into temporary pointer
B557 8623 3217 STX Z23
B559 E430 3218 BB559 CPX Z30

```

B55B	D004	3219	BNE BB561	;if end of name table
B55D	C52F	3220	CMP Z2F	;not reached yet
B55F	F005	3221	BEQ BB566	
B561	20BDB5	3222	BB561 JSR SB5BD	;find highest text area
B564	F0F3	3223	BEQ BB559	
B566	8558	3224	BB566 STA Z58	;save pointer into array area
B568	8659	3225	STX Z59	
B56A	A903	3226	LDA \$03	;reset step to 3
B56C	8553	3227	STA Z53	
B56E	A558	3228	BB56E LDA Z58	;get pointer to next array
B570	A659	3229	LDX Z59	
B572	E432	3230	BB572 CPX Z32	
B574	D007	3231	BNE BB57D	;if end of array area
B576	C531	3232	CMP Z31	
B578	D003	3233	BNE BB57D	
B57A	4C06B6	3234	JMP JB606	;move text area
B57D	8522	3235	BB57D STA Z22	;else save array pointer
B57F	8623	3236	STX Z23	;into temporary pointer
B581	A000	3237	LDY \$00	
B583	B122	3238	LDA (Z22),Y	
B585	AA	3239	TAX	;save first character of name
B586	C8	3240	INY	
B587	B122	3241	LDA (Z22),Y	
B589	08	3242	PHP	;save status of second character of name
B58A	C8	3243	INY	
B58B	B122	3244	LDA (Z22),Y	;add length of array
B58D	6558	3245	ADC Z58	
B58F	8558	3246	STA Z58	;to array pointer
B591	C8	3247	INY	
B592	B122	3248	LDA (Z22),Y	
B594	6559	3249	ADC Z59	;high byte also
B596	8559	3250	STA Z59	
B598	28	3251	PLP	;second character of variable name
B599	10D3	3252	BPL BB56E	;must have bit 7 set for string
B59B	8A	3253	TXA	;first character of variable name
B59C	30D0	3254	BMI BB56E	;must have bit 7 low for string
B59E	C8	3255	INY	
B59F	B122	3256	LDA (Z22),Y	;get # of dimensions
B5A1	A000	3257	LDY \$00	;and initialize Y
B5A3	0A	3258	ASL A	;compute offset to first element
B5A4	6905	3259	ADC \$05	
B5A6	6522	3260	ADC Z22	
B5A8	8522	3261	STA Z22	;point to first element
B5AA	9002	3262	BCC BB5AE	
B5AC	E623	3263	INC Z23	
B5AE	A623	3264	BB5AE LDX Z23	
B5B0	E459	3265	BB5B0 CPX Z59	;if not pointing beyond last element
B5B2	D004	3266	BNE BB5B8	
B5B4	C558	3267	CMP Z58	
B5B6	F0BA	3268	BEQ BB572	
B5B8	20C7B5	3269	BB5B8 JSR SB5C7	;find highest text address
B5BB	F0F3	3270	BEQ BB5B0	;and repeat for next address


```

3272 ;check string pointed to by Z22/Z23
3273 ;as being highest string area outside allocated area
3274 ;
B5BD B122 3275 SB5BD LDA (Z22),Y ;if name has bit 7 of first char = 0
B5BF 3035 3276 BMI BB5F6
B5C1 C8 3277 INY
B5C2 B122 3278 LDA (Z22),Y ;and bit 7 of second character = 1
B5C4 1030 3279 BPL BB5F6
B5C6 C8 3280 INY ;drop thru
3281 ;
3282 ;check string indexed by descriptor
3283 ;for being highest string area
3284 ;outside of allocated area
3285 ;
B5C7 B122 3286 SB5C7 LDA (Z22),Y ;if length = 0
B5C9 F02B 3287 BEQ BB5F6 ;then no text area
B5CB C8 3288 INY
B5CC B122 3289 LDA (Z22),Y
B5CE AA 3290 TAX ;get pointer to text into AX
B5CF C8 3291 INY
B5D0 B122 3292 LDA (Z22),Y
B5D2 C534 3293 CMP Z34 ;if pointer is
B5D4 9006 3294 BCC BB5DC ;above string storage area
B5D6 D01E 3295 BNE BB5F6
B5D8 E433 3296 CPX Z33
B5DA B01A 3297 BCS BB5F6 ;then string is not highest
B5DC C560 3298 BB5DC CMP Z60 ;if pointer is
B5DE 9016 3299 BCC BB5F6 ;below previous highest
B5E0 D004 3300 BNE BB5E6
B5E2 E45F 3301 CPX Z5F
B5E4 9010 3302 BCC BB5F6 ;then not highest
B5E6 865F 3303 BB5E6 STX Z5F
B5E8 8560 3304 STA Z60 ;save pointer to new highest
B5EA A522 3305 LDA Z22
B5EC A623 3306 LDX Z23 ;get pointer to variable or descriptor
B5EE 854E 3307 STA Z4E
B5F0 864F 3308 STX Z4F ;save pointer to winner
B5F2 A553 3309 LDA Z53 ;save step size of winner
B5F4 8555 3310 STA Z55
B5F6 A553 3311 BB5F6 LDA Z53 ;add step
B5F8 18 3312 CLC
B5F9 6522 3313 ADC Z22 ;to variable or descriptor pointer
B5FB 8522 3314 STA Z22
B5FD 9002 3315 BCC BB601
B5FF E623 3316 INC Z23
B601 A623 3317 BB601 LDX Z23 ;variable or descriptor pointer in AX
B603 A000 3318 LDY $00
B605 60 3319 RTS

```

```

3321 ;continuation of string garbage clean-up
3322 ;
B606 A54F 3323 JB606 LDA Z4F      ;if no pointer to descriptor
B608 054E 3324      ORA Z4E      ;of highest string yet,
B60A F0F5 3325      BEQ BB601    ;return
B60C A555 3326      LDA Z55
B60E 2904 3327      AND $04      ;compute offset to descriptor
B610 4A    3328      LSR A
B611 A8    3329      TAY
B612 8555 3330      STA Z55
B614 B14E 3331      LDA (Z4E),Y  ;get length of text
B616 655F 3332      ADC Z5F      ;add to pointer to beginning of text
B618 855A 3333      STA Z5A      ;store in high limit for move routine
B61A A560 3334      LDA Z60      ;high bytes also
B61C 6900 3335      ADC $00
B61E 855B 3336      STA Z5B
B620 A533 3337      LDA Z33      ;move current string storage pointer
B622 A634 3338      LDX Z34
B624 8558 3339      STA Z58      ;into high limit of output for move
B626 8659 3340      STX Z59
B628 20BFA3 3341     JSR SA3BF    ;move bytes
B62B A455 3342      LDY Z55      ;get offset to descriptor
B62D C8    3343      INY
B62E A558 3344      LDA Z58
B630 914E 3345      STA (Z4E),Y  ;let descriptor point to moved text area
B632 AA    3346      TAX
B633 E659 3347      INC Z59
B635 A559 3348      LDA Z59
B637 C8    3349      INY
B638 914E 3350      STA (Z4E),Y  ;high byte also
B63A 4C2AB5 3351     JMP JB52A   ;repeat for next string

```

```

3353 ;diadic operator "+" for strings
3354 ;
B63D A565 3355 JB63D LDA Z65
B63F 48 3356 PHA ;save pointer to left operand
B640 A564 3357 LDA Z64
B642 48 3358 PHA
B643 2083AE 3359 JSR SAE83 ;get value of next operand in flp accu
B646 208FAD 3360 JSR SAD8F ;must be string
B649 68 3361 PLA
B64A 856F 3362 STA Z6F
B64C 68 3363 PLA
B64D 8570 3364 STA Z70 ;restore temporary string address
B64F A000 3365 LDY $00
B651 B16F 3366 LDA (Z6F),Y ;get length of left operand
B653 18 3367 CLC
B654 7164 3368 ADC (Z64),Y ;add length of right operand
B656 9005 3369 BCC BB65D ;if => 256
B658 A217 3370 LDY $17 ;point to STRING TOO LONG Error
B65A 4C37A4 3371 JMP JA437 ;print error
3372 ;
B65D 2075B4 3373 BB65D JSR SB475 ;allocate area for sum of lengths
B660 207AB6 3374 JSR SB67A ;move left operand
B663 A550 3375 LDA Z50
B665 A451 3376 LDY Z51 ;get descriptor of right operand
B667 1AC025 3377 JSR SB6AA ;de-allocate old area
B66A 208CB6 3378 JSR SB68C ;move right operand
B66D A56F 3379 LDA Z6F
B66F A470 3380 LDY Z70
B671 20AAB6 3381 JSR SB6AA ;de-allocate old area
B674 20CAB4 3382 JSR BB4CA ;save descriptor on stack
B677 4CB8AD 3383 JMP JADB8 ;go back to expression evaluation

```

```

3385 ;move string with descriptor pointed by Z6F/Z70
3386 ;into last allocated area
3387 ;
B67A A000 3388 SB67A LDY $00
B67C B16F 3389 LDA (Z6F),Y
B67E 48 3390 PHA ;save length from descriptor
B67F C8 3391 INY
B680 B16F 3392 LDA (Z6F),Y
B682 AA 3393 TAX ;pointer from descriptor
B683 C8 3394 INY
B684 B16F 3395 LDA (Z6F),Y
B686 A8 3396 TAY ;into XY
B687 68 3397 PLA ;length into A
3398 ;
3399 ;move string with length in A, pointer in XY
3400 ;into last allocated area
3401 ;
B688 8622 3402 SB688 STX Z22 ;save pointer in temporary pointer area
B68A 8423 3403 STY Z23
B68C A8 3404 SB68C TAY ;if length = 0
B68D F00A 3405 BEQ BB699 ;don't move
B68F 48 3406 PHA ;save length
B690 88 3407 BB690 DEY ;for every character
B691 B122 3408 LDA (Z22),Y ;move from string
B693 9135 3409 STA (Z35),Y ;to last allocated area
B695 98 3410 TYA
B696 D0F8 3411 BNE BB690 ;until no more characters
B698 68 3412 PLA ;restore length
B699 18 3413 BB699 CLC ;set Z35/Z36 to end of allocated area
B69A 6535 3414 ADC Z35
B69C 8535 3415 STA Z35
B69E 9002 3416 BCC BB6A2
B6A0 E636 3417 INC Z36
B6A2 60 3418 BB6A2 RTS

```

```

3420 ;de-allocate a temporary string
3421 ;
B6A3 208FAD 3422 SB6A3 JSR SAD8F      ;check if value is a string
B6A6 A564 3423 SB6A6 LDA Z64
B6A8 A465 3424 LDY Z65      ;set AY to pointer to descriptor
B6AA 8522 3425 SB6AA STA Z22      ;and store in temporary pointer
B6AC 8423 3426 STY Z23
B6AE 20DBB6 3427 JSR SB6DB      ;check descriptor stack
B6B1 08 3428 PHP              ;save status (Z=1 when desc on stack)
B6B2 A000 3429 LDY $00
B6B4 B122 3430 LDA (Z22),Y
B6B6 48 3431 PHA              ;save string length
B6B7 C8 3432 INY
B6B8 B122 3433 LDA (Z22),Y
B6BA AA 3434 TAX              ;pointer to string in Y
B6BB C8 3435 INY
B6BC B122 3436 LDA (Z22),Y
B6BE A8 3437 TAY
B6BF 68 3438 PLA              ;string length into A
B6C0 28 3439 PLP              ;restore status
B6C1 D013 3440 BNE BB6D6      ;branch if descriptor on stack
B6C3 C434 3441 CPY Z34        ;if pointer to string
B6C5 D00F 3442 BNE BB6D6
B6C7 E433 3443 CPX Z33        ;= string storage pointer
B6C9 D00B 3444 BNE BB6D6
B6CB 48 3445 PHA              ;save length
B6CC 18 3446 CLC
B6CD 6533 3447 ADC Z33        ;add length to string storage pointer
B6CF 8533 3448 STA Z33        ;to form new string storage pointer
B6D1 9002 3449 BCC BB6D5      ;(de-allocates string)
B6D3 E634 3450 INC Z34
B6D5 68 3451 BB6D5 PLA        ;restore length
B6D6 8622 3452 BB6D6 STX Z22   ;save pointer to string
B6D8 8423 3453 STY Z23
B6DA 60 3454 RTS
3455 ;
3456 ;check descriptor stack
3457 ;
B6DB C418 3458 SB6DB CPY Z18
B6DD D00C 3459 BNE BB6EB      ;if descriptor pointer
B6DF C517 3460 CMP Z17        ;same as previous descriptor stack ptr
B6E1 D008 3461 BNE BB6EB
B6E3 8516 3462 STA Z16        ;then set descriptor stack index
B6E5 E903 3463 SBC $03        ;to point to it
B6E7 8517 3464 STA Z17        ;set previous descriptor index below it
B6E9 A000 3465 LDY $00        ;leave new descriptor pointer in AY
B6EB 60 3466 BB6EB RTS

```

```

3468 ;"CHR$" command
3469 ;
B6EC 20A1B7 3470 WB6EC JSR SB7A1 ;get integer parameter into X
B6EF 8A 3471 TXA
B6FO 48 3472 PHA ;and save
B6F1 A901 3473 LDA $01 ;set length
B6F3 207DB4 3474 JSR SB47D ;allocate area
B6F6 68 3475 PLA ;restore integer
B6F7 A000 3476 LDY $00
B6F9 9162 3477 STA (Z62),Y ;save character in allocated area
B6FB 68 3478 PLA
B6FC 68 3479 PLA ;remove own return address
B6FD 4CCAB4 3480 JMP BB4CA ;save descriptor on descriptor stack
3481 ;
3482 ;"LEFT$" command
3483 ;
B700 2061B7 3484 WB700 JSR SB761 ;get 2 parameters
B703 D150 3485 CMP (Z50),Y ;compare length wanted to stack length
B705 98 3486 TYA ;A = initial index in text
B706 9004 3487 JB706 BCC BB70C ;if length requested >= string length
B708 B150 3488 LDA (Z50),Y
B70A AA 3489 TAX ;use string length instead
B70B 98 3490 TYA
B70C 48 3491 BB70C PHA ;save initial index
B70D 8A 3492 BB70D TXA ;get length wanted
B70E 48 3493 BB70E PHA ;and save it
B70F 207DB4 3494 JSR SB47D ;de-allocate area
B712 A550 3495 LDA Z50
B714 A451 3496 LDY Z51 ;set AY to point to first parameter
B716 20AAB6 3497 JSR SB6AA ;de-allocate temporary string
B719 68 3498 PLA ;restore length requested
B71A A8 3499 TAY
B71B 68 3500 PLA ;restore initial index in text
B71C 18 3501 CLC
B71D 6522 3502 ADC Z22 ;add to text pointer
B71F 8522 3503 STA Z22
B721 9002 3504 BCC BB725
B723 E623 3505 INC Z23
B725 98 3506 BB725 TYA ;for length requested,
B726 208CB6 3507 JSR SB68C ;move string into allocated area
B729 4CCAB4 3508 JMP BB4CA ;save descriptor on descriptor stack
3509 ;
3510 ;"RIGHT$" command
3511 ;
B72C 2061B7 3512 WB72C JSR SB761 ;get 2 parameters
B72F 18 3513 CLC
B730 F150 3514 SBC (Z50),Y ;compute length - length requested
B732 49FF 3515 EOR $FF
B734 4C06B7 3516 JMP JB706 ;go to tail end of LEFT$

```

```

3518 ;"MID$" command
3519 ;
B737 A9FF 3520 WB737 LDA $FF ;set default 3rd parameter to 255
B739 8565 3521 STA Z65
B73B 207900 3522 JSR X0079 ;get current character
B73E C929 3523 CMP ")" ;if not ")"
B740 F006 3524 BEQ BB748
B742 20FDAE 3525 JSR SAEFD ;must be ","
B745 209EB7 3526 JSR SB79E ;get next integer
B748 2061B7 3527 BB748 JSR SB761 ;get first 2 parameters
B74B F04B 3528 BEQ BB798 ;if 2nd parameter = 0, ILLEGAL QUANTITY
B74D CA 3529 DEX ;decrement index
B74E 8A 3530 TXA
B74F 48 3531 PHA ;and save it on stack
B750 18 3532 CLC
B751 A200 3533 LDX $00
B753 F150 3534 SBC (Z50),Y ;compute index - length
B755 B0B6 3535 BCS BB70D ;if larger, get null string
B757 49FF 3536 EOR $FF ;if # characters remaining
B759 C565 3537 CMP Z65
B75B 90B1 3538 BCC BB70E ;less than 3rd parameter,
B75D A565 3539 LDA Z65 ;take # of characters remaining instead
B75F B0AD 3540 BCS BB70E ;do tail end of LEFT$
3541 ;
3542 ;get first two parameters for LEFT$, RIGHT$ and MID$
3543 ;
B761 20F7AE 3544 SB761 JSR SAEF7 ;next character must be ")"
B764 68 3545 PLA
B765 A f 3546 JAL ;save return address
B766 68 3547 PLA
B767 8555 3548 STA Z55 ;remove return address
B769 68 3549 PLA ;restore 2nd parameter (integer)
B76A 68 3550 PLA
B76B 68 3551 PLA
B76C AA 3552 TAX
B76D 68 3553 PLA
B76E 8550 3554 STA Z50 ;move 1st parameter (ptr to descriptor)
B770 68 3555 PLA ;into Z50/Z51
B771 8551 3556 STA Z51
B773 A555 3557 LDA Z55
B775 48 3558 PHA ;restore return address
B776 98 3559 TYA
B777 48 3560 PHA
B778 A000 3561 LDY $00
B77A 8A 3562 TXA ;set flags according to 2nd parameter
B77B 60 3563 RTS

```

```

3565 ;"LEN$" command
3566 ;
B77C 2082B7 3567 WB77C JSR SB782 ;de-allocate parameter, get length in Y
B77F 4CA2B3 3568 JMP JB3A2 ;convert integer in Y to flp
3569 ;
B782 20A3B6 3570 SB782 JSR SB6A3 ;de-allocate temp string (parameter)
B785 A200 3571 LDX $00
B787 860D 3572 STX Z0D ;reset string flag
B789 A8 3573 TAY ;move length to Y
B78A 60 3574 RTS
3575 ;
3576 ;"ASC$" command
3577 ;
B78B 2082B7 3578 WB78B JSR SB782 ;de-allocate parameter, get length in Y
B78E F008 3579 BEQ BB798 ;if null string, ILLEGAL QUANTITY Error
B790 A000 3580 LDY $00
B792 B122 3581 LDA (Z22),Y ;get first character of string
B794 A8 3582 TAY ;move into Y
B795 4CA2B3 3583 JMP JB3A2 ;convert integer in Y to flp
3584 ;
B798 4C48B2 3585 BB798 JMP JB248 ; print error
3586 ;
3587 ;fetch integer value in X and check range
3588 ;
B79B 207300 3589 SB79B JSR X0073 ;get next character
B79E 208AAD 3590 SB79E JSR SAD8A ;get next non-string value
B7A1 20B8B1 3591 SB7A1 JSR SB1B8 ;convert positive flp accu to integer
B7A4 A664 3592 LDX Z64 ;if not in range 0 - 255
B7A6 D0F0 3593 BNE BB798 ;print error ILLEGAL QUANTITY
B7A8 A665 3594 LDX Z65 ;load value into X
B7AA 4C7900 3595 JMP X0079 ;get current character again and return

```



```

3597 ;"VAL" command
3598 ;
B7AD 2082B7 3599 WB7AD JSR SB782 ;de-allocate parameter, get length in Y
B7B0 D003 3600 BNE BB7B5 ;if length zero,
B7E2 4CF7B8 3601 JMP JB8F7 ;set result zero
B7B5 A67A 3602 BB7B5 LDX Z7A ;move current character pointer
B7B7 A47B 3603 LDY Z7B
B7B9 8671 3604 STX Z71 ;to temporary pointer
B7BB 8472 3605 STY Z72
B7BD A622 3606 LDX Z22 ;move low byte of text pointer
B7BF 867A 3607 STX Z7A ;into current character pointer
B7C1 18 3608 CLC
B7C2 6522 3609 ADC Z22 ;add length
B7C4 8524 3610 STA Z24 ;to create pointer to end of string
B7C6 A623 3611 LDX Z23 ;move high byte of text pointer
B7C8 867B 3612 STX Z7B ;into high byte of current character ptr
B7CA 9001 3613 BCC BB7CD
B7CC E8 3614 INX ;add carry of length
B7CD 8625 3615 BB7CD STX Z25 ;set high byte of ptr to end of string
B7CF A000 3616 LDY $00
B7D1 B124 3617 LDA (Z24),Y ;get first byte beyond string
B7D3 48 3618 PHA ;and save it on stack
B7D4 98 3619 TYA
B7D5 9124 3620 STA (Z24),Y ;set null instead
B7D7 207900 3621 JSR X0079 ;get current character
B7DA 20F3BC 3622 JSR SBCF3 ;convert string to flp
B7DD 68 3623 PLA ;restore first byte beyond string
B7DE A000 3624 LDY $00
B7E0 9124 3625 STA (Z24),Y ;and restore
B7E2 A671 3626 SB7E2 LDX Z71 ;get new value for current character ptr
B7E4 A472 3627 LDY Z72
B7E6 867A 3628 STX Z7A ;and store it as current character ptr
B7E8 847B 3629 STY Z7B
B7EA 60 3630 RTS
3631 ;
3632 ;get address into Z14/Z15 and integer into X
3633 ;
B7EB 208AAD 3634 SB7EB JSR SAD8A ;get non-string value from statement
B7EE 20F7B7 3635 JSR SB7F7 ;convert flp accu to integer at Z14/Z15
B7F1 20FDAE 3636 SB7F1 JSR SAEFD ;next character must be ", "
B7F4 4C9EB7 3637 JMP SB79E ;get next integer from statement into X

```

```

3639 ;convert floating point into integer in Z14/Z15
3640 ;
B7F7 A566 3641 SB7F7 LDA Z66 ;if negative
B7F9 309D 3642 BMI BB798
B7FB A561 3643 LDA Z61
B7FD C991 3644 CMP $91 ;or exponent > 16
B7FF B097 3645 BCS BB798 ;then ILLEGAL QUANTITY Error
B801 209BBC 3646 JSR SBC9B ;convert flp accu to integer at Z64/Z65
B804 A564 3647 LDA Z64
B806 A465 3648 LDY Z65
B808 8414 3649 STY Z14 ;and into result field
B80A 8515 3650 STA Z15
B80C 60 3651 RTS
3652 ;
3653 ;"PEEK" command
3654 ;
B80D A515 3655 WB80D LDA Z15
B80F 48 3656 PHA
B810 A514 3657 LDA Z14 ;save contents of Z14/Z15
B812 48 3658 PHA
B813 20F7B7 3659 JSR SB7F7 ;convert flp accu to integer in Z14/Z15
B816 A000 3660 LDY $00
B818 B114 3661 LDA (Z14),Y ;get byte
B81A A8 3662 TAY ;and save in Y
B81B 68 3663 PLA ;then restore Z14/Z15
B81C 8514 3664 STA Z14
B81E 68 3665 PLA
B81F 8515 3666 STA Z15
B821 4CA2B3 3667 JMP JB3A2 ;convert integer in Y to flp
3668 ;
3669 ;"POKE" command
3670 ;
B824 20EBB7 3671 WB824 JSR SB7EB ;get address in Z14/Z15, integer in X
B827 8A 3672 TXA
B828 A000 3673 LDY $00
B82A 9114 3674 STA (Z14),Y ;move integer from X to address
B82C 60 3675 RTS

```

```

3677 ;"WAIT" command
3678 ;
B82D 20EBB7 3679 WB82D JSR SB7EB ;get address in Z14/Z15, integer in X
B830 8649 3680 STX Z49 ;save mask
B832 A200 3681 LDX $00 ;default pattern = 00
B834 207900 3682 JSR X0079 ;get current character
B837 F003 3683 BEQ BB83C ;if not end of statement
B839 20F1B7 3684 JSR SB7F1 ;get next integer from statement into X
B83C 864A 3685 BB83C STX Z4A ;save pattern
B83E A000 3686 LDY $00
B840 B114 3687 BB840 LDA (Z14),Y ;get byte
B842 454A 3688 EOR Z4A ;Exclusive OR with pattern
B844 2549 3689 AND Z49 ;mask
B846 F0F8 3690 BEQ BB840 ;repeat until non-zero
B848 60 3691 BB848 RTS
3692 ;
3693 ;add 0.5 to flp accu (half rounding)
3694 ;
B849 A911 3695 SB849 LDA <TBFl1
B84B A0BF 3696 LDY >TBFl1 ;set AY to point to value 0.5
B84D 4C67B8 3697 JMP SB867 ;add to flp accu
3698 ;
3699 ;"MINUS" operator
3700 ;
B850 208CBA 3701 SB850 JSR SB8BC ;subtract flp accu from # indexed by AY
3702 ;
3703 ;diadic operator "-"
3704 ;
B853 A566 3705 WB853 LDA Z66
B855 49FF 3706 EOR $FF ;complement sign
B857 8566 3707 STA Z66
B859 456E 3708 EOR Z6E
B85B 856F 3709 STA Z6F ;also adjust Exclusive OR of both signs
B85D A561 3710 LDA Z61 ;fetch exponent of flp accu
B85F 4C6AB8 3711 JMP JB86A ;go apply diadic operator "+"

```

```

B862 2099B9 3713 BB862 JSR SB999      ;perform preshifts according to A
B865 903C   3714      BCC BB8A3      ;do addition or subtraction
          3715 ;
          3716 ;add flp # indexed by AY to flp accu
          3717 ;
B867 208CBA 3718 SB867 JSR SBA8C      ;perform addition
          3719 ;
          3720 ;"PLUS" operator
          3721 ;
B86A        3722 WB86A = *
B86A D003   3723 JB86A BNE BB86F      ;if flp accu = 0
B86C 4CFCBB 3724      JMP SBBFC      ;move 2nd flp accu into flp accu
          3725 ;
B86F A670   3726 BB86F LDX Z70        ;save guard bit
B871 8656   3727      STX Z56
B873 A269   3728      LDX $69        ;X = pointer to smaller
B875 A569   3729      LDA Z69        ;get exponent of 2nd flp accu
B877 A8      3730 SB877 TAY          ;if zero
B878 FOCE   3731      BEQ BB848      ;result already in flp accu
B87A 38      3732      SEC
B87E E561   3733      SBC Z61        ;compute difference in exponents
B87D FO24   3734      BEQ BB8A3      ;if zero, go act on fractions
B87F 9012   3735      BCC BB893      ;if second flp accu higher
B881 8461   3736      STY Z61        ;set exponent of result
B883 A46E   3737      LDY Z6E        ;and sign of result
B885 8466   3738      STY Z66
B887 49FF   3739      EOR $FF
B889 6900   3740      ADC $00        ;two's complement of difference
B88B A000   3741      LDY $00
B88D 8456   3742      STY Z56        ;set guard bit to 0 for higher
B88F A261   3743      LDX $61        ;point to smaller
B891 D004   3744      BNE BB897
B893 A000   3745 BB893 LDY $00        ;if second flp accu lower
B895 8470   3746      STY Z70        ;move 0 to guard bit of lower
B897 C9F9   3747 BB897 CMP $F9      ;if difference more than 8
B899 30C7   3748      BMI BB862      ;do preshifts
B89B A8      3749      TAY          ;save number of preshifts
B89C A570   3750      LDA Z70        ;get guard bit
B89E 5601   3751      LSR Z01,X      ;pad with zero
B8A0 20B0B9 3752      JSR SB9B0      ;do preshifts
B8A3 246F   3753 BB8A3 BIT Z6F      ;if both signs are the same
B8A5 1057   3754      BPL BB8FE      ;add fractions

```

```

3756 ;negate flp accu if borrow, and postshift
3757 ;
B8A7 A061 3758 LDY $61 ;let Y point
B8A9 E069 3759 CFX $69
B8AB F002 3760 BEQ B88AF
B8AD A069 3761 LDY $69 ;to higher
B8AF 38 3762 B88AF SEC
B8B0 49FF 3763 EOR $FF ;compute initial borrow
B8B2 6556 3764 ADC Z56 ;from guard bit
B8B4 8570 3765 STA Z70
B8B6 B90400 3766 LDA Z04,Y
B8B9 F504 3767 SBC Z04,X ;compute difference, byte 4
B8BB 8565 3768 STA Z65
B8BD B90300 3769 LDA Z03,Y
B8C0 F503 3770 SBC Z03,X ;byte 3
B8C2 8564 3771 STA Z64
B8C4 B90200 3772 LDA Z02,Y
B8C7 F502 3773 SBC Z02,X ;byte 2
B8C9 8563 3774 STA Z63
B8CB B90100 3775 LDA Z01,Y
B8CE F501 3776 SBC Z01,X ;and byte 1
B8D0 8562 3777 STA Z62
B8D2 B003 3778 JB8D2 BCS B88D7 ;if borrow
B8D4 2047B9 3779 JSR SB947 ;negate result
B8D7 A000 3780 B88D7 LDY $00 ;initialize parameters
B8D9 98 3781 TYA
B8DA 18 3782 CLC ;for postshift
B8DB A662 3783 B88DB LDX Z62 ;if most significant byte 0
B8DD D04A 3784 BNE B8929
B8DF A663 3785 LDX Z63 ;shift fraction over one byte
B8E1 8662 3786 STX Z62
B8E3 A664 3787 LDX Z64
B8E5 8663 3788 STX Z63
B8E7 A665 3789 LDX Z65
B8E9 8664 3790 STX Z64
B8EB A670 3791 LDX Z70
B8ED 8665 3792 STX Z65
B8EF 8470 3793 STY Z70
B8F1 6908 3794 ADC $08 ;correct exponent by 8
B8F3 C920 3795 CMP $20
B8F5 D0E4 3796 BNE B88DB ;repeat until 4 bytes shifted
B8F7 A900 3797 JB8F7 LDA $00 ;because then result = 0
B8F9 8561 3798 JB8F9 STA Z61
B8FB 8566 3799 JB8FB STA Z66
B8FD 60 3800 RTS

```

```

3802 ;add fractions
3803 ;
B8FE 6556 3804 BB8FE ADC Z56 ;compute initial fraction
B900 8570 3805 STA Z70 ;also set guard bit
B902 A565 3806 LDA Z65
B904 656D 3807 ADC Z6D ;compute sum, byte 4
B906 8565 3808 STA Z65
B908 A564 3809 LDA Z64
B90A 656C 3810 ADC Z6C ;byte 3
B90C 8564 3811 STA Z64
B90E A563 3812 LDA Z63
B910 656B 3813 ADC Z6B ;byte 2
B912 8563 3814 STA Z63
B914 A562 3815 LDA Z62
B916 656A 3816 ADC Z6A ;byte 1
B918 8562 3817 STA Z62
B91A 4C36B9 3818 JMP JB936 ;go adjust for carry outside byte 1
3819 ;
3820 ;postshift
3821 ;
B91D 6901 3822 BB91D ADC $01 ;add 1 to exponent
B91F 0670 3823 ASL Z70 ;shift left 1 bit
B921 2665 3824 ROL Z65 ;over whole fraction
B923 2664 3825 ROL Z64
B925 2663 3826 ROL Z63
B927 2662 3827 ROL Z62
B929 10F2 3828 BB929 BPL BB91D ;repeat until MSB = 1
B92B 38 3829 SEC ;subtract correction from exponent
B92C E561 3830 SBC Z61 ;underflow causes zero result
B92E B0C7 3831 BCS JB8F7
B930 49FF 3832 EOR $FF
B932 6901 3833 ADC $01 ;store new exponent
B934 8561 3834 STA Z61 ;carry means overflow in fraction
B936 900E 3835 JB936 BCC BB946 ;so increment exponent
B938 E661 3836 JB938 INC Z61 ;unless exponent overflows
B93A F042 3837 BEQ BB97E ;and shift right by one bit
B93C 6662 3838 ROR Z62
B93E 6663 3839 ROR Z63
B940 6664 3840 ROR Z64
B942 6665 3841 ROR Z65
B944 6670 3842 ROR Z70 ;including guard bit
B946 60 3843 BB946 RTS

```

```

3845 ;negate flp accu
3846 ;
B947 A566 3847 SB947 LDA Z66
B949 49FF 3848 EOR $FF ;complement sign
B94B 8566 3849 STA Z66
B94D A562 3850 SB94D LDA Z62
B94F 49FF 3851 EOR $FF ;complement fraction byte 4
B951 8562 3852 STA Z62
B953 A563 3853 LDA Z63
B955 49FF 3854 EOR $FF ;byte 3
B957 8563 3855 STA Z63
B959 A564 3856 LDA Z64
B95B 49FF 3857 EOR $FF ;byte 2
B95D 8564 3858 STA Z64
B95F A565 3859 LDA Z65
B961 49FF 3860 EOR $FF ;and byte 1
B963 8565 3861 STA Z65
B965 A570 3862 LDA Z70
B967 49FF 3863 EOR $FF ;and guard bit
B969 8570 3864 STA Z70 ;if guard bit 0
B96B E670 3865 INC Z70
B96D D00E 3866 BNE BB97D ;then increment fraction
3867 ;
3868 ;increment fraction
3869 ;
B96F E665 3870 SB96F INC Z65 ;increment byte 4
B971 D00A 3871 BNE BB97D ;if 0
B973 E664 3872 INC Z64 ;byte 3 also
B975 D006 3873 BNE BB97D ;if 0
B977 E663 3874 INC Z63 ;byte 2 also
B979 D002 3875 BNE BB97D ;if 0
B97B E662 3876 INC Z62 ;byte 1 also
B97D 60 3877 BB97D RTS ;zero here means overflow
3878 ;
B97E A20F 3879 BB97E LDX $0F ;point to OVERFLOW Error
B980 4C37A4 3880 JMP JA437 ;print message

```

	3882	;preshift	
	3883	;	
B983 A225	3884	JB983 LDX \$25	
B985 B404	3885	BB985 LDY Z04,X	;shift whole fraction
B987 8470	3886	STY Z70	;right one byte
B989 B403	3887	LDY Z03,X	
B98B 9404	3888	STY Z04,X	
B98D B402	3889	LDY Z02,X	
B98F 9403	3890	STY Z03,X	
B991 B401	3891	LDY Z01,X	
B993 9402	3892	STY Z02,X	
B995 A468	3893	LDY Z68	;high order padding
B997 9401	3894	STY Z01,X	
B999 6908	3895	SB999 ADC \$08	;add 8 to exponent
B99B 30E8	3896	BMI BB985	;if still <= 0
B99D F0E6	3897	BEQ BB985	;shift another byte
B99F E908	3898	SBC \$08	;compensate for correction
B9A1 A8	3899	TAY	
B9A2 A570	3900	LDA Z70	;check guard bit
B9A4 B014	3901	BCS BB9BA	;if exponent zero now, stop shifting
B9A6 1601	3902	BB9A6 ASL Z01,X	;else
B9A8 9002	3903	BCC BB9AC	
B9AA F601	3904	INC Z01,X	
B9AC 7601	3905	BB9AC ROR Z01,X	
B9AE 7601	3906	ROR Z01,X	;shift fraction right one bit
B9B0 7602	3907	SB9B0 ROR Z02,X	
B9B2 7603	3908	ROR Z03,X	
B9B4 7604	3909	ROR Z04,X	
B9B6 6A	3910	ROR A	;last bit into A
B9B7 C8	3911	INY	
B9B8 DOEC	3912	BNE BB9A6	;until # of shifts zero
B9BA 18	3913	BB9BA CLC	
B9BB 60	3914	RTS	


```

3916 ;1, also used as default FOR step
B9BC 810000 3917 TB9BC .BY $81,$00,$00,$00,$00
3918 ;polynome table for LOG
B9C1 03 3919 TB9C1 .BY $03
3920 ;0.434255942
B9C2 7F5E56 3921 .BY $7F,$5E,$56,$CB,$79
3922 ;0.576584541
B9C7 80139B 3923 .BY $80,$13,$9B,$0B,$64
3924 ;0.961800759
B9CC 807638 3925 .BY $80,$76,$38,$93,$16
3926 ;2.88539007
B9D1 8238AA 3927 .BY $82,$38,$AA,$3B,$20
3928 ;0.5 * SQR(2)
B9D6 803504 3929 TB9D6 .BY $80,$35,$04,$F3,$34
3930 ;SQR(2)
B9DB 813504 3931 TB9DB .BY $81,$35,$04,$F3,$34
3932 ;-0.5
B9E0 808000 3933 TB9E0 .BY $80,$80,$00,$00,$00
3934 ;LOG(2)
B9E5 803172 3935 TB9E5 .BY $80,$31,$72,$17,$F8
3936 ;
3937 ;"LOG" command
3938 ;
B9EA 3939 WB9EA = *
B9EA 202B8C 3940 SB9EA JSR SBC2B ;get sign of flp accu into A
B9ED F002 3941 BEQ BB9F1
B9EF 1003 3942 BPL BB9F4 ;if > 0, OK
B9F1 4C48E2 3943 BB9F1 JMP JB248 ;else ILLEGAL QUANTITY Error
3944 ;
B9F4 A561 3945 BB9F4 LDA Z61 ;get exponent
B9F6 E97F 3946 SBC $7F ;correct for excess of 128
B9F8 48 3947 PBA ;and save result
B9F9 A980 3948 LDA $80
B9FB 8561 3949 STA Z61 ;set exponent
B9FD A9D6 3950 LDA <TB9D6
B9FF A0B9 3951 LDY >TB9D6 ;set AY to point to 0.5 * SQR(2)
BA01 2067B8 3952 JSR SB867 ;add to flp accu
BA04 A9DB 3953 LDA <TB9DB
BA06 A0B9 3954 LDY >TB9DB ;set AY to point to SQR(2)
BA08 200FBB 3955 JSR SBBOF ;divide AY by flp accu
BA0B A9BC 3956 LDA <TB9BC
BA0D A0B9 3957 LDY >TB9BC ;set AY to point to 1
BA0F 2050B8 3958 JSR SB850 ;subtract flp accu from AY
BA12 A9C1 3959 LDA <TB9C1
BA14 A0B9 3960 LDY >TB9C1 ;set AY to polynome table
BA16 2043E0 3961 JSR XE043 ;compute odd polynome
BA19 A9E0 3962 LDA <TB9E0
BA1B A0B9 3963 LDY >TB9E0 ;set AY to point to -0.5
BA1D 2067B8 3964 JSR SB867 ;add to flp accu
BA20 68 3965 PLA ;restore exponent
BA21 207EBD 3966 JSR JBD7E ;add exponent to flp accu
BA24 A9E5 3967 LDA <TB9E5
BA26 A0B9 3968 LDY >TB9E5 ;set AY to LOG(2)
BA28 208CBA 3969 SBA28 JSR SBA8C ;multiply AY times flp accu

```

	3971	;"*" operator	
	3972	;	
BA2B D003	3973	WBA2B BNE BBA30	;if flp accu = 0, return
BA2D 4C8BBA	3974	JMP JBA8B	
	3975	;	
BA30 20B7BA	3976	BBA30 JSR SBAB7	;add exponents
BA33 A900	3977	LDA \$00	
BA35 8526	3978	STA Z26	;set flp accu extension to zero
BA37 8527	3979	STA Z27	
BA39 8528	3980	STA Z28	
BA3B 8529	3981	STA Z29	
BA3D A570	3982	LDA Z70	;get guard bit
BA3F 2059BA	3983	JSR SBA59	* second flp accu added to flp accu ext
BA42 A565	3984	LDA Z65	
BA44 2059BA	3985	JSR SBA59	;same for byte 4
BA47 A564	3986	LDA Z64	
BA49 2059BA	3987	JSR SBA59	;same for byte 3
BA4C A563	3988	LDA Z63	
BA4E 2059BA	3989	JSR SBA59	;and byte 2
BA51 A562	3990	LDA Z62	
BA53 205EBA	3991	JSR SBA5E	;and byte 1
BA56 4C8FBB	3992	JMP JBB8F	;post-normalize and return
	3993	;	
	3994	;add second flp accu * A to flp extension	
	3995	;	
BA59 D003	3996	SBA59 BNE SBA5E	;if a zero
BA5B 4C83B9	3997	JMP JB983	;perform pre-shift
	3998	;	
BA5E 4A	3999	SBA5E LSR A	;shift LSB of A into carry
BA5F 0980	4000	ORA \$80	;set to \$80 only after 8 shifts
BA61 A8	4001	BBA61 TAY	;save A
BA62 9019	4002	ECC BBA7D	;if low bit set
BA64 18	4003	CLC	
BA65 A529	4004	LDA Z29	
BA67 656D	4005	ADC Z6D	;add second flp fraction
BA69 8529	4006	STA Z29	;to flp extension
BA6B A528	4007	LDA Z28	
BA6D 656C	4008	ADC Z6C	
BA6F 8528	4009	STA Z28	;also byte 2
BA71 A527	4010	LDA Z27	
BA73 656B	4011	ADC Z6B	
BA75 8527	4012	STA Z27	;and byte 3
BA77 A526	4013	LDA Z26	
BA79 656A	4014	ADC Z6A	
BA7B 8526	4015	STA Z26	;and byte 4
BA7D 6626	4016	BBA7D ROR Z26	;shift flp extension right 1 bit
BA7F 6627	4017	ROR Z27	
BA81 6628	4018	ROR Z28	
BA83 6629	4019	ROR Z29	
BA85 6670	4020	ROR Z70	;including guard bit
BA87 98	4021	TYA	;restore A
BA88 4A	4022	LSR A	;get low bit of A into carry
BA89 D0D6	4023	BNE BBA61	;repeat until zero
BA8B 60	4024	JBA8B RTS	

	4026	;move flp # indexed by AY into second flp accu	
	4027	;	
BA8C 8522	4028	SBA8C STA Z22	;save AY index
BA8E 8423	4029	STY Z23	
BA90 A004	4030	LDY \$04	
BA92 B122	4031	LDA (Z22),Y	;move byte 4 of fraction
BA94 856D	4032	STA Z6D	
BA96 88	4033	DEY	
BA97 B122	4034	LDA (Z22),Y	
BA99 856C	4035	STA Z6C	;byte 3
BA9B 88	4036	DEY	
BA9C B122	4037	LDA (Z22),Y	
BA9E 856B	4038	STA Z6B	;byte 2
BAA0 88	4039	DEY	
BAA1 B122	4040	LDA (Z22),Y	
BAA3 856E	4041	STA Z6E	;and byte 1
BAA5 4566	4042	EOR Z66	;into sign
BAA7 856F	4043	STA Z6F	;also Exclusive OR with sign of flp accu
BAA9 A56E	4044	LDA Z6E	
BAAB 0980	4045	ORA \$80	
BAAD 856A	4046	STA Z6A	;byte 1 always has bit 7 set
BAAF 88	4047	DEY	
BAB0 B122	4048	LDA (Z22),Y	
BAB2 8569	4049	STA Z69	;save exponent
BAB4 A561	4050	LDA Z61	;show exponent of flp accu in A, Z and N
BAB6 60	4051	RTS	

```

4053 ;add exponents
4054 ;
BAB7 A569 4055 SBAB7 LDA Z69 ;get exp. of 2nd flp accu (left operand)
BAB9 F01F 4056 BEQ BBADA ;if zero, set zero result
BAB8 18 4057 CLC
BAB8 6561 4058 ADC Z61 ;add to exponent of flp accu
BABE 9004 4059 BCC BBAC4 ;if carry set
BAC0 301D 4060 BMI BBADF ;and negative, true overflow
BAC2 18 4061 CLC ;else correct for excess 128
BAC3 2C 4062 .BY $2C ;skip next instruction
BAC4 1014 4063 BBAC4 BPL BBADA ;if C = 1 and positive, real underflow
BAC6 6980 4064 ADC $80 ;else correct for excess 128
BAC8 8561 4065 STA Z61 ;store resulting exponent
BACA D003 4066 BNE BBACF ;if zero
BACC 4CFB8 4067 JMP JB8FB ;set sign positive
4068 ;
BACF A56F 4069 BBACF LDA Z6F ;Exclusive OR of signs
BAD1 8566 4070 STA Z66 ;is sign of result
BAD3 60 4071 RTS
4072 ;
BAD4 A566 4073 LDA Z66 ;get sign of flp accu
BAD6 49FF 4074 .EOR $FF
BAD8 3005 4075 BMI BBADF ;if positive, overflow
BADA 68 4076 BBADA PLA ;remove own return address
BADB 68 4077 PLA
BADC 4CF7B8 4078 JMP JB8F7 ;and set result zero
4079 ;
BADF 4C7EB9 4080 BBADF JMP BB97E ;OVERFLOW Error
4081 ;
4082 ;multiply flp accu by 10
4083 ;
BAE2 200CBC 4084 SBAE2 JSR SBCOC ;move rounded flp into second flp accu
BAE5 AA 4085 TAX ;get exponent
BAE6 F010 4086 BEQ BBAF8 ;if flp accu is zero, return
BAE8 18 4087 CLC
BAE9 6902 4088 ADC $02 ;exponent of second flp + 2 (# * 4)
BAEB B0F2 4089 BCS BBADF ;exit if overflow
BAED A200 4090 SBAED LDX $00
BAEF 866F 4091 STX Z6F ;set signs same
BAF1 2077B8 4092 JSR SB877 ;add second flp accu to flp (# * 5)
BAF4 E661 4093 INC Z61 ;add 1 to exponent (# * 10)
BAF6 F0E7 4094 BEQ BBADF ;exit if overflow
BAF8 60 4095 BBAF8 RTS
4096 ;
4097 ;flp constant 10 for division
BAF9 842000 4098 TBAF9 .BY $84,$20,$00,$00,$00
4099 ;
4100 ;divide flp accu by 10
4101 ;
BAFE 200CBC 4102 SBAFE JSR SBCOC ;move rounded flp accu to 2nd flp accu
BB01 A9F9 4103 LDA <TBAF9
BB03 A0BA 4104 LDY >TBAF9 ;set AY to flp value 10
BB05 A200 4105 LDX $00
BB07 866F 4106 STX Z6F ;set signs the same
BB09 20A2BB 4107 JSR SBBA2 ;load flp accu from AY
BB0C 4C12BB 4108 JMP JBB12 ;perform division

```

```

4110 ;divide number (indexed by AY) by flp accu
4111 ;
BBOF 208CBA 4112 SBBOF JSR SBA8C ;fetch # indexedby AY into 2nd flp accu
4113 ;
4114 ;"/" operator
4115 ;
BB12 4116 WBB12 = . *
BB12 F076 4117 JBB12 BEQ BBB8A ;if flp accu = 0, DIVISION BY ZERO Error
BB14 201BBC 4118 JSR SBC1B ;round flp accu according to guard bit
BB17 A900 4119 LDA $00
BB19 38 4120 SEC
BB1A E561 4121 SBC Z61 ;compute 2's complement of exponent
BB1C 8561 4122 STA Z61
BB1E 20B7BA 4123 JSR SBAB7 ;add exponents
BB21 E661 4124 INC Z61 ;correct exponent
BB23 F0BA 4125 BEQ BBADF ;exit if overflow
BB25 A2FC 4126 LDX $FC ;set initial index into flp extension
BB27 A901 4127 LDA $01 ;assure 8 loops per byte
BB29 A46A 4128 BBB29 LDY Z6A
BB2B C462 4129 CPY Z62 ;compare fractions, byte 1
BB2D D010 4130 BNE BBB3F
BB2F A46B 4131 LDY Z6B
BB31 C463 4132 CPY Z63 ;byte 2
BB33 D00A 4133 BNE BBB3F
BB35 A46C 4134 LDY Z6C
BB37 C464 4135 CPY Z64 ;byte 3
BB39 D004 4136 BNE BBB3F
BB3B A46D 4137 LDY Z6D
BB3D C465 4138 CPY Z65 ;byte 4
BB3F 08 4139 BBB3F PHP ;save result of comparison
BB40 2A 4140 ROL A ;shift result into A
BB41 9009 4141 BCC BBB4C ;after 8 shifts
BB43 E8 4142 INX
BB44 9529 4143 STA Z29,X ;save result into flp extension
BB46 F032 4144 BEQ BBB7A ;after byte 4, do guard byte
BB48 1034 4145 BPL BBB7E ;after guard byte, exit
BB4A A901 4146 LDA $01 ;again, 8 iterations per byte
BB4C 28 4147 BBB4C PLP ;restore result of comparison
BB4D B00E 4148 BCS BBB5D ;if dividend larger, subtract
BB4F 066D 4149 JBB4F ASL Z6D ;shift dividend left 1 bit
BB51 266C 4150 ROL Z6C
BB53 266B 4151 ROL Z6B
BB55 266A 4152 ROL Z6A
BB57 B0E6 4153 BCS BBB3F ;if C = 1 comparison result will be 1
BB59 30CE 4154 BMI BBB29 ;if C = 0 and MSB = 1, compare
BB5B 10E2 4155 BPL BBB3F ;if C = 0 and MSB = 0, comp. result = 0
BB5D A8 4156 BBB5D TAY ;if dividend larger
BB5E A56D 4157 LDA Z6D
BB60 E565 4158 SBC Z65 ;subtract divisor
BB62 856D 4159 STA Z6D
BB64 A56C 4160 LDA Z6C
BB66 E564 4161 SBC Z64 ;byte 2
BB68 856C 4162 STA Z6C
BB6A A56B 4163 LDA Z6B
BB6C E563 4164 SBC Z63 ;byte 3
BB6E 856B 4165 STA Z6B
BB70 A56A 4166 LDA Z6A
BB72 E562 4167 SBC Z62 ;byte 4
BB74 856A 4168 STA Z6A
BB76 98 4169 TYA

```

```

BB77 4C4FBB 4170      JMP JBB4F      ;repeat
                        4171 ;
BB7A A940 4172 BBB7A LDA $40          ;two more iterations for guard bit
BB7C DOCE 4173      BNE BBB4C
BB7E 0A 4174 BBB7E ASL A              ;after guard byte computation,
BB7F 0A 4175      ASL A              ;shift result left 6 bits
BB80 0A 4176      ASL A
BB81 0A 4177      ASL A
BB82 0A 4178      ASL A
BB83 0A 4179      ASL A
BB84 8570 4180      STA Z70          ;set new guard bit
BB86 28 4181      PLP                ;remove result of last comparison
BB87 4C8FBB 4182     JMP JBB8F      ;and save result
                        4183 ;
BB8A A214 4184 BBB8A LDX $14          ;point to DIVISION BY ZERO Error
BB8C 4C37A4 4185     JMP JA437       ;print error
                        4186 ;
BB8F A526 4187 JBB8F LDA Z26          ;move flp accu extension
BB91 8562 4188      STA Z62          ;into flp fraction
BB93 A527 4189      LDA Z27
BB95 8563 4190      STA Z63          ;byte 2
BB97 A528 4191      LDA Z28
BB99 8564 4192      STA Z64          ;byte 3
BB9B A529 4193      LDA Z29
BB9D 8565 4194      STA Z65          ;byte 4
BB9F 4CD7B8 4195     JMP BB8D7       ;do post-normalization
                        4196 ;
                        4197 ;load flp accu with constant indexed by AY
                        4198 ;
BBAA 8522 4199 SBBAA STA Z22
BBA4 8423 4200      STY Z23          ;save AY
BBA6 A004 4201      LDY $04
BBA8 B122 4202      LDA (Z22),Y     ;load byte + 4
BBA A 8565 4203      STA Z65          ;into flp accu + 4
BBAC 88 4204      DEY
BBAD B122 4205      LDA (Z22),Y     ;byte + 3
BBAF 8564 4206      STA Z64
BBB1 88 4207      DEY
BBB2 B122 4208      LDA (Z22),Y     ;byte + 2
BBB4 8563 4209      STA Z63
BBB6 88 4210      DEY
BBB7 B122 4211      LDA (Z22),Y     ;byte + 1
BBB9 8566 4212      STA Z66          ;into sign byte
BBB B 0980 4213      ORA $80          ;and with bit 7 set
BBBD 8562 4214      STA Z62          ;into flp accu + 1
BBBF 88 4215      DEY
BBCO B122 4216      LDA (Z22),Y     ;byte + 0
BBC2 8561 4217      STA Z61          ;into exponent of flp accu
BBC4 8470 4218      STY Z70          ;clear guard bit
BBC6 60 4219      RTS

```

```

4221 ;store flp accu at Z5C-Z60
4222 ;
BBC7 A25C 4223 LDX <Z5C ;initialize low byte of pointer
BBC9 2C 4224 .BY $2C ;skip next instruction
4225 ;
4226 ;store flp accu at Z57-Z5B
4227 ;
BBCA A257 4228 LDX <Z57 ;initialize low byte of pointer
BBCC A000 4229 LDY >Z57 ;initialize high byte of pointer
BBCE F004 4230 BEQ SBBDD4 ;JMP
4231 ;
4232 ;store flp accu into area indexed by Z49/Z4A
4233 ;
BBDO A649 4234 JBBDO LDX Z49 ;set area pointer in XY
BBD2 A44A 4235 LDY Z4A
4236 ;
4237 ;store flp accu in area indexed by XY
4238 ;
BBD4 201BBC 4239 SBBDD4 JSR SBC1B ;round flp accu
BBD7 8622 4240 STX Z22 ;store XY in temporary pointer
BBD9 8423 4241 STY Z23
BBD8 A004 4242 LDY $04
BBD0 A565 4243 LDA Z65 ;move byte 4 of fraction
BBD7 9122 4244 STA (Z22),Y
BBE1 88 4245 DEY
BBE2 A564 4246 LDA Z64
BBE4 9122 4247 STA (Z22),Y ;byte 3
BBE6 88 4248 DEY
BBE7 A563 4249 LDA Z63
BBE9 9122 4250 STA (Z22),Y ;byte 2
BBEB 88 4251 DEY
BBEC A566 4252 LDA Z66 ;combine sign
BBEE 097F 4253 ORA $7F
BBF0 2562 4254 AND Z62 ;and byte 1
BBF2 9122 4255 STA (Z22),Y ;into byte 1 of area
BBF4 88 4256 DEY
BBF5 A561 4257 LDA Z61 ;move exponent
BBF7 9122 4258 STA (Z22),Y
BBF9 8470 4259 STY Z70 ;clear guard bit
BBFB 60 4260 RTS
4261 ;
4262 ;move second flp accu into first flp accu
4263 ;
BBFC A56E 4264 SBBFC LDA Z6E ;move sign
BBFE 8566 4265 SBBFE STA Z66
BC00 A205 4266 LDX $05 ;loop counter
BC02 B568 4267 BBC02 LDA Z68,X ;move 5 bytes
BC04 9560 4268 STA Z60,X
BC06 CA 4269 DEX
BC07 D0F9 4270 BNE BBC02
BC09 8670 4271 STX Z70 ;set product of signs to "+"
BC0B 60 4272 RTS

```

```

4274 ;move rounded flp accu into second flp accu
4275 ;
BC0C 201BBC 4276 SBC0C JSR SBC1B ;round flp accu
BC0F A206 4277 LDX $06 ;set # bytes to move
BC11 B560 4278 BBC11 LDA Z60,X ;move 6 bytes
BC13 9568 4279 STA Z68,X
BC15 CA 4280 DEX
BC16 D0F9 4281 BNE BBC11
BC18 8670 4282 STX Z70 ;store zero into guard bit
BC1A 60 4283 BBC1A RTS
4284 ;
4285 ;round flp accu according to guard bit
4286 ;
BC1B A561 4287 SBC1B LDA Z61 ;if flp accu = 0
BC1D F0FB 4288 BEQ BBC1A ;leave it at 0
BC1F 0670 4289 ASL Z70 ;if guard bit not set
BC21 90F7 4290 BCC BBC1A ;status quo
BC23 206FB9 4291 SBC23 JSR SB96F ;increment fraction
BC26 D0F2 4292 BNE BBC1A ;if no overflow, leave it
BC28 4C38B9 4293 JMP JB938 ;else shift fraction right 1 bit
4294 ;
4295 ;get sign of flp accu in A
4296 ;
BC2B A561 4297 SBC2B LDA Z61 ;if exponent zero
BC2D F009 4298 BEQ BBC38 ;sign = 0
BC2F A566 4299 BBC2F LDA Z66 ;move sign bit
BC31 2A 4300 JBC31 ROL A ;into C bit
BC32 A9FF 4301 LDA $FF ;if C = 1, sign = FF
BC34 B002 4302 BCS BBC38
BC36 A901 4303 LDA $01 ;if C = 0, sign = 01
BC38 60 4304 BBC38 RTS
4305 ;
4306 ;"SGN" command
4307 ;
BC39 202BBC 4308 WBC39 JSR SBC2B ;get SGN of flp accu into A
4309 ;
4310 ;move signed number from A into flp accu
4311 ;
BC3C 8562 4312 JBC3C STA Z62 ;store # in most significant byte
BC3E A900 4313 LDA $00
BC40 8563 4314 STA Z63 ;and 0 into byte 2
BC42 A288 4315 LDX $88 ;load exponent
BC44 A562 4316 JBC44 LDA Z62
BC46 49FF 4317 EOR $FF ;complement sign
BC48 2A 4318 ROL A ;and move into C flag
BC49 A900 4319 SBC49 LDA $00
BC4B 8565 4320 STA Z65 ;clear byte 3
BC4D 8564 4321 STA Z64 ;and byte 4
BC4F 8661 4322 JBC4F STX Z61 ;store exponent
BC51 8570 4323 STA Z70 ;move sign to guard bit
BC53 8566 4324 STA Z66 ;and to sign bit
BC55 4CD2B8 4325 JMP JB8D2 ;negate flp accu if borrow and post-shift

```



```

4327 ;"ABS" command
4328 ;
BC58 4666 4329 WBC58 LSR Z66 ;set sign of flp accu positive
BC5A 60 4330 RTS
4331 ;
4332 ;compare flp accu to flp # indexed by AY
4333 ;
BC5B 8524 4334 SBC5B STA Z24 ;save index to variable
BC5D 8425 4335 SBC5D STY Z25
BC5F A000 4336 LDY $00
BC61 B124 4337 LDA (Z24),Y ;get exponent of variable
BC63 C8 4338 INY
BC64 AA 4339 TAX
BC65 F0C4 4340 BEQ SBC2B ;if zero, get sign of flp accu
BC67 B124 4341 LDA (Z24),Y ;get byte 1 of fraction
BC69 4566 4342 EOR Z66 ;if different from sign of flp accu
BC6B 30C2 4343 BMI BBC2F ;get sign of flp accu
BC6D E461 4344 CPX Z61 ;compare exponents
BC6F D021 4345 BNE BBC92 ;if not equal, set sign according to C
BC71 B124 4346 LDA (Z24),Y
BC73 0980 4347 ORA $80 ;get byte 1
BC75 C562 4348 CMP Z62 ;compare to byte 1 of flp accu
BC77 D019 4349 BNE BBC92 ;if not equal, set sign according to C
BC79 C8 4350 INY
BC7A B124 4351 LDA (Z24),Y ;get byte 2
BC7C C563 4352 CMP Z63 ;compare to byte 2 of flp accu
BC7E D012 4353 BNE BBC92 ;if not equal, set sign according to C
BC80 C8 4354 INY
BC81 B124 4355 LDA (Z24),Y ;get byte 3
BC83 C564 4356 CMP Z64 ;compare to byte 3 of flp accu
BC85 D00B 4357 BNE BBC92 ;if not equal, set sign according to C
BC87 C8 4358 INY
BC88 A97F 4359 LDA $7F ;set borrow
BC8A C570 4360 CMP Z70 ;according to guard bit
BC8C B124 4361 LDA (Z24),Y
BC8E E565 4362 SBC Z65 ;compute difference between bytes 4
BC90 F028 4363 BEQ BBCBA ;if equal, return with A = 0
BC92 A566 4364 BBC92 LDA Z66 ;else compute sign according to C
BC94 9002 4365 BCC BBC98
BC96 49FF 4366 EOR $FF
BC98 4C31BC 4367 BBC98 JMP JBC31 ;and set A accordingly

```

```

4369 ;convert flp accu to a 4 byte signed integer
4370 ;
BC9B A561 4371 SBC9B LDA Z61 ;get exponent
BC9D F04A 4372 BEQ BBCE9 ;if zero fill fraction with zeros
BC9F 38 4373 SEC
BCA0 E9A0 4374 SBC $A0 ;calculate exponent - 32
BCA2 2466 4375 BIT Z66 ;if result negative
BCA4 1009 4376 BPL BBCAF
BCA6 AA 4377 TAX ;save A
BCA7 A9FF 4378 LDA $FF ;set padding to ones
BCA9 8568 4379 STA Z68
BCAB 204DB9 4380 JSR SB94D ;negate flp fraction
BCAE 8A 4381 TXA ;restore A
BCAF A261 4382 BBCAF LDX $61
BCB1 C9F9 4383 CMP $F9 ;if < 8 places to be shifted
BCB3 1006 4384 BPL BBCEB
BCB5 2099B9 4385 JSR SB999 ;do shift
BCB8 8468 4386 STY Z68 ;and reset padding
BCBA 60 4387 BBCBA RTS
4388 ;
BCBB A8 4389 BBCBB TAY ;if > = 8 places to be shifted
BCBC A566 4390 LDA Z66
BCBE 2980 4391 AND $80
BCC0 4662 4392 LSR Z62 ;set parameters
BCC2 0562 4393 ORA Z62
BCC4 8562 4394 STA Z62 ;and MSB
BCC6 20B0B9 4395 JSR SB9B0 ;to do byte shifts first
BCC9 8468 4396 STY Z68 ;and then reset padding
BCCB 60 4397 RTS
4398 ;
4399 ;"INT" command
4400 ;
BCCC 4401 WBCCC = *
BCCC A561 4402 SBCCC LDA Z61 ;get exponent
BCCE C9A0 4403 CMP $A0 ;if > = 32
BCD0 B020 4404 BCS BBCF2 ;then no conversion necessary
BCD2 209BBC 4405 JSR SBC9B ;convert flp accu to 4 byte integer
BCD5 8470 4406 STY Z70 ;clear guard bit
BCD7 A566 4407 LDA Z66 ;get sign
BCD9 8466 4408 STY Z66 ;set sign positive
BCDB 4980 4409 EOR $80 ;invert sign
BCDD 2A 4410 ROL A ;and move into carry
BCDE A9A0 4411 LDA $A0
BCE0 8561 4412 STA Z61 ;set exponent to 32
BCE2 A565 4413 LDA Z65 ;move least significant bit
BCE4 8507 4414 STA Z07 ;into even/odd switch
BCE6 4CD2B8 4415 JMP JB8D2 ;negate flp accu if borrow
4416 ;
4417 ;clear flp accu
4418 ;
BC9E 8562 4419 BBCE9 STA Z62 ;move A into every byte of fraction
BCEB 8563 4420 STA Z63
BCED 8564 4421 STA Z64
BCEF 8565 4422 STA Z65
BCF1 A8 4423 TAY
BCF2 60 4424 BBCF2 RTS

```

```

4426 ;convert string to flp # in flp accu
4427 ;
BCF3 A000 4428 SBCF3 LDY $00
BCF5 A20A 4429 LDX $0A ;loop counter, 11 bytes to be cleared
BCF7 945D 4430 BCF7 STY Z5D,X ;clear flp accu and 5 more
BCF9 CA 4431 DEX
BCFA 10FB 4432 BPL BCF7
BCFC 900F 4433 BCC BBD0D ;if first character not numeric
BCFE C92D 4434 CMP '-' ;check for minus
BD00 D004 4435 BNE BBD06 ;if so,
BD02 8667 4436 STX Z67 ;set flag for negative #
BD04 F004 4437 BEQ BBD0A
BD06 C92B 4438 BBD06 CMP '+' ;check for positive number
BD08 D005 4439 BNE BBD0F ;if not, go try others
BD0A 207300 4440 BBD0A JSR X0073 ;get next character
BD0D 905B 4441 BBD0D BCC BBD6A ;if numeric, gather number
BD0F C92E 4442 BBD0F CMP '.' ;if decimal point,
BD11 F02E 4443 BEQ BBD41 ;set flag
BD13 C945 4444 CMP 'E' ;if exponent character
BD15 D030 4445 BNE BBD47
BD17 207300 4446 JSR X0073 ;get next character
BD1A 9017 4447 BCC BBD33 ;if numeric, gather exponent value
BD1C C9AB 4448 CMP $AB ;if code for "--"
BD1E F00E 4449 BEQ BBD2E
BD20 C92D 4450 CMP '-' ;or "--" itself
BD22 F00A 4451 BEQ BBD2E ;set flag
BD24 C9AA 4452 CMP $AA ;if code for "+"
BD26 F008 4453 BEQ BBD30
BD28 C92B 4454 CMP '+' ;or "+" itself
BD2A F004 4455 BEQ BBD30 ;skip
BD2C D007 4456 BNE BBD35 ;else combine fraction & exponent
BD2E 6660 4457 BBD2E ROR Z60 ;set flag for negative exponent
BD30 207300 4458 BBD30 JSR X0073 ;get next character
BD33 905C 4459 BBD33 BCC BBD91 ;if numeric, gather exponent value
BD35 2460 4460 BBD35 BIT Z60 ;at end of exponent
BD37 100E 4461 BPL BBD47 ;if sign of exponent negative
BD39 A900 4462 LDA $00
BD3B 38 4463 SEC
BD3C E55E 4464 SBC Z5E ;get two's complement of exponent
BD3E 4C49BD 4465 JMP JBD49 ;combine
4466 ;
BD41 665F 4467 BBD41 ROR Z5F ;set code for "." found
BD43 245F 4468 BIT Z5F
BD45 50C3 4469 BVC BBD0A ;if already set, this is end of number
BD47 A55E 4470 BBD47 LDA Z5E ;get exponent, base 10
BD49 38 4471 JBD49 SEC
BD4A E55D 4472 SBC Z5D ;minus # of digits after "."
BD4C 855E 4473 STA Z5E ;save
BD4E F012 4474 BEQ BBD62 ;if zero, no adjustment
BD50 1009 4475 BPL BBD5B ;if positive, multiply by 10
BD52 20FEBA 4476 BBD52 JSR SBAFE ;if negative, divide flp accu by 10
BD55 E65E 4477 INC Z5E ;increment exponent, base 10
BD57 D0F9 4478 BNE BBD52 ;and repeat if not zero
BD59 F007 4479 BEQ BBD62
BD5B 20E2BA 4480 BBD5B JSR SBAE2 ;if positive, multiply flp accu by 10
BD5E C65E 4481 DEC Z5E ;decrement exponent, base 10
BD60 D0F9 4482 BNE BBD5B ;and repeat if not zero
BD62 A567 4483 BBD62 LDA Z67 ;get sign of flp #
BD64 3001 4484 BMI BBD67 ;if positive,
BD66 60 4485 RTS ;return

```

```

4486 ;
BD67 4CB4BF 4487 BBD67 JMP JBFB4 ;else apply monadic "-"
4488 ;
BD6A 48 4489 BBD6A PHA ;save numeric digit
BD6B 245F 4490 BIT Z5F ;if "." found
BD6D 1002 4491 BPL BBD71
BD6F E65D 4492 INC Z5D ;increment # of digits after "."
BD71 20E2BA 4493 BBD71 JSR SBAE2 ;multiply flp accu by 10
BD74 68 4494 PLA ;restore numeric digit
BD75 38 4495 SEC
BD76 E930 4496 SBC '0 ;convert from ASCII to binary
BD78 207EBD 4497 JSR JBD7E ;add signed integer from A to flp accu
BD7E 4C0ABD 4498 JMP BBD0A ;repeat for next character
4499 ;
4500 ;add signed integer from A to flp accu
4501 ;
BD7E 48 4502 JBD7E PHA ;save integer
BD7F 200CBC 4503 JSR SBC0C ;move rounded flp accu to 2nd flp accu
BD82 68 4504 PLA ;restore integer
BD83 203CBC 4505 JSR JBC3C ;get signed # from A into flp accu
BD86 A56E 4506 LDA Z6E
BD88 A566 4507 EOR Z66 ;compute Exclusive OR of signs
BD8A 856F 4508 STA Z6F
BD8C A661 4509 LDX Z61 ;let Z and N reflect exponent
BD8E 4C6AB8 4510 JMP JB86A ;apply diadic operator "+"
4511 ;
4512 ;get exponent of number from a string
4513 ;
BD91 A55E 4514 BBD91 LDA Z5E ;get exponent gathered so far
BD93 C90A 4515 CMP $0A ;if => 10
BD95 9009 4516 BCC BBD40
BD97 A964 4517 LDA $64 ;and sign of exponent is positive
BD99 2460 4518 BIT Z60
BD9B 3011 4519 BMI BBD4E ;then overflow
BD9D 4C7EB9 4520 JMP BB97E ;print error
4521 ;
BDA0 0A 4522 BBD40 ASL A ;exponent * 2
BDA1 0A 4523 ASL A ;exponent * 4
BDA2 18 4524 CLC
BDA3 655E 4525 ADC Z5E ;exponent * 5
BDA5 0A 4526 ASL A ;exponent * 10
BDA6 18 4527 CLC
BDA7 A000 4528 LDY $00
BDA9 717A 4529 ADC (Z7A),Y ;add current character
BDAB 38 4530 SEC
BDAC E930 4531 SBC '0 ;convert from ASCII to binary
BDAE 855E 4532 BBD4E STA Z5E ;save exponent
BDB0 4C30BD 4533 JMP BBD30 ;repeat for next character

```

```

4535 ;constants for flp to string conversion
4536 ;
4537 ;999999.9
BDB3 9B3EBC 4538 TDB3 .BY $9B,$3E,$BC,$1F,$FD
4539 ;999999999
BDB8 9E6E6B 4540 TDB8 .BY $9E,$6E,$6B,$27,$FD
4541 ;1000000000
BDBD 9E6E6B 4542 TDBD .BY $9E,$6E,$6B,$28,$00
4543 ;
4544 ;print IN follwed by statement #
4545 ;
BDC2 A971 4546 SBDC2 LDA <TA371 ;set AY to IN message
BDC4 A0A3 4547 LDY >TA371
BDC6 20DABD 4548 JSR SBDDA ;print message
BDC9 A53A 4549 LDA Z3A ;get current statement # into AX
BDCB A639 4550 LDX Z39
4551 ;
4552 ;print # from AX
4553 ;
BDCD 8562 4554 SBDCD STA Z62 ;move AX into flp accu
BDCF 8663 4555 STX Z63
BDD1 A290 4556 LDX $90 ;set exponent to 16
BDD3 38 4557 SEC
BDD4 2049BC 4558 JSR SBC49 ;pad out flp accu
BDD7 20DFBD 4559 JSR SBDDF ;convert flp accu to string
BDDA 4C1EAB 4560 SBDDA JMP SABL1E ;and go print string
4561 ;
4562 ;convert # in flp accu to string
4563 ;
BDDD A001 4564 SBDDD LDY $01 ;initial output index
BDDF A920 4565 SBDDF LDA $20 ;set first character to a space
BDE1 2466 4566 BIT Z66 ;but if sign negative
BDE3 1002 4567 BPL BBDE7
BDE5 A92D 4568 LDA "-" ;set first character to "-"
BDE7 99FF00 4569 BBDE7 STA X0100-1,Y ;store sign into output area
BDEA 8566 4570 STA Z66 ;set sign positive
BDEC 8471 4571 STY Z71 ;save output index
BDEE C8 4572 INY
BDEF A930 4573 LDA "0" ;assume 0
BDF1 A661 4574 LDX Z61 ;if exponent 0
BDF3 D003 4575 BNE BBDF8
BDF5 4C04BF 4576 JMP JBFO4 ;output 0 and end string
4577 ;
BDF8 A900 4578 BBDF8 LDA $00 ;clear exponent, base 10
BDFA E080 4579 CPX $80
BDFC F002 4580 BEQ BBEO0 ;if exponent, base 2, = 0
BDFF B009 4581 BCS BBEO9 ;or < 0
BE00 A9BD 4582 BBEO0 LDA <TDBD
BE02 A0BD 4583 LDY >TDBD
BE04 2028BA 4584 JSR SBA28 ;multiply by flp accu
BE07 A9F7 4585 LDA $F7 ;set exponent base 10 to -9
BE09 855D 4586 BBEO9 STA Z5D ;save exponent base 10
BE0B A9B8 4587 BBEOB LDA <TBDB3
BE0D A0BD 4588 LDY >TBDB8 ;set AY to index 999999999
BE0F 205BBC 4589 JSR SBC5B ;compare flp accu to AY
BE12 F01E 4590 BEQ BBEO32 ;if equal, no need to adjust exonent
BE14 1012 4591 BPL BBEO28 ;if flp accu >, adjust accordingly
BE16 A9B3 4592 BBEO16 LDA <TBDB3 ;if flp accu <,
BE18 A0BD 4593 LDY >TBDB3 ;set AY to point to 999999.9
BE1A 205BBC 4594 JSR SBC5B ;and compare flp accu to # indexed by AY

```

BE1D	F002	4595	BEQ	BBE21	
BE1F	100E	4596	BPL	BBE2F	;if flp accu <, no more adjustment
BE21	20E2BA	4597	BBE21	JSR	SBAE2
BE24	C65D	4598	DEC	Z5D	;else multiply flp accu by 10
BE26	D0EE	4599	BNE	BBE16	;decrement exponent base 10
BE28	20FEBA	4600	BBE28	JSR	SBAFE
BE2B	E65D	4601	INC	Z5D	;and repeat
BE2D	D0DC	4602	BNE	BBE0B	;divide flp accu by 10
BE2F	2049B8	4603	BBE2F	JSR	SB849
BE32	209BEC	4604	BBE32	JSR	SBC9B
BE35	A201	4605	LDX	\$01	;increment exponent, base 10
BE37	A55D	4606	LDA	Z5D	;repeat
BE39	18	4607	CLC		;do half rounding of flp accu
BE3A	690A	4608	ADC	\$0A	;convert flp accu to 4 byte integer
BE3C	3009	4609	BMI	BBE47	;set position of decimal point
BE3E	C90B	4610	CMF	\$0B	;get exponent, base 10
BE40	B006	4611	BCS	BBE48	
BE42	69FF	4612	ADC	\$FF	;add 10
BE44	AA	4613	TAX		;if still < 0, do exponential notation
BE45	A902	4614	LDA	\$02	;if > 10
BE47	38	4615	BBE47	SEC	;do exponential notation
BE48	E902	4616	BBE48	SBC	\$02
BE4A	855E	4617	STA	Z5E	;correct exponent, base 10
BE4C	865D	4618	STX	Z5D	;save exponent
BE4E	8A	4619	TXA		;save position of decimal point
BE4F	F002	4620	BEQ	BBE53	
BE51	1013	4621	BPL	BBE66	;if position of decimal point > 0
BE53	A471	4622	BBE53	LDY	Z71
BE55	A92E	4623	LDA	"	;no extra adjustment needed
BE57	C8	4624	INY		;get output index
BE58	99FF00	4625	STA	X0100-1,Y	;set decimal point
BE5B	8A	4626	TXA		;store it
BE5C	F006	4627	BEQ	BBE64	;if position of decimal point < 0
BE5E	A930	4628	LDA	"0	
BE60	C8	4629	INY		;set a 0
BE61	99FF00	4630	STA	X0100-1,Y	;and store it
BE64	8471	4631	BBE64	STY	Z71
BE66	A000	4632	BBE66	LDY	\$00
BE68	A280	4633	SBE68	LDX	\$80
BE6A	A565	4634	BBE6A	LDA	Z65
BE6C	18	4635	CLC		;save output index
BE6D	7919BF	4636	ADC	TBF16+3,Y	;set initial table index
BE70	8565	4637	STA	Z65	;initial table phase negative
BE72	A564	4638	LDA	Z64	;add table entry
BE74	7918BF	4639	ADC	TBF16+2,Y	
BE77	8564	4640	STA	Z64	
BE79	A563	4641	LDA	Z63	
BE7B	7917BF	4642	ADC	TBF16+1,Y	;byte 2
BE7E	8563	4643	STA	Z63	
BE80	A562	4644	LDA	Z62	
BE82	7916BF	4645	ADC	TBF16,Y	;byte 1
BE85	8562	4646	STA	Z62	
BE87	E8	4647	INX		;increment digit
BE88	B004	4648	BCS	BBE8E	
BE8A	10DE	4649	BPL	BBE6A	;if no overflow and positive, repeat
BE8C	3002	4650	BMI	BBE90	
BE8E	30DA	4651	BBE8E	BMI	BBE6A
BE90	8A	4652	BBE90	TXA	;if overflow and negative phase, repeat
BE91	9004	4653	BCC	BBE97	;if positive phase
BE93	49FF	4654	EOR	\$FF	;get 10 - digit

BE95	690A	4655	ADC \$0A	
BE97	692F	4656	BBE97 ADC \$2F	;convert to ASCII
BE99	C8	4657	INY	
BE9A	C8	4658	INY	
BE9B	C8	4659	INY	
BE9C	C8	4660	INY	;point to next table entry
BE9D	8447	4661	STY Z47	;save table index
BE9F	A471	4662	LDY Z71	;get output index
BEA1	C8	4663	INY	
BEA2	AA	4664	TAX	;save digit
BEA3	297F	4665	AND \$7F	;remove phase bit
BEA5	99FF00	4666	STA X0100-1,Y	;store digit
BEA8	C65D	4667	DEC Z5D	;decrement position of decimal point
BEAA	D006	4668	BNE BBEB2	;if position reached
BEAC	A92E	4669	LDA ^.	;set decimal point
BEAE	C8	4670	INY	
BEAF	99FF00	4671	STA X0100-1,Y	;and store it at next output location
BEB2	8471	4672	BBEB2 STY Z71	;save output index
BEB4	A447	4673	LDY Z47	;restore table index
BEB6	8A	4674	TXA	
BEB7	49FF	4675	EOR \$FF	;inverse phase bit
BEB9	2980	4676	AND \$80	;reset digit
BEBB	AA	4677	TAX	
BEBC	C024	4678	CPY \$24	;if table index not at 9 (* 4)
BEBE	F004	4679	BEQ BBEC4	
BEC0	C03C	4680	CPY \$3C	;and not at 15 (* 4)
BEC2	D0A6	4681	BNE BBEE6A	;repeat
BEC4	A471	4682	BBEC4 LDY Z71	;if end of segment, restore output index
BEC6	B9FF00	4683	BBEC6 LDA X0100-1,Y	;get last digit
BEC9	88	4684	DEY	
BECA	C930	4685	CMP ^0	;if a "0"
BECC	F0F8	4686	BEQ BBEC6	;remove and repeat
BECE	C92E	4687	CMP ^.	;if decimal point
BED0	F001	4688	BEQ BBED3	;remove, but do not repeat
BED2	C8	4689	INY	
BED3	A92B	4690	BBED3 LDA ^+	;set "+"
BED5	A65E	4691	LDX Z5E	;get exponent, base 10
BED7	F02E	4692	BEQ BBFO7	;if zero, go end string
BED9	1008	4693	BPL BBEE3	
BEDE	A900	4694	LDA \$00	;if negative,
BEDD	38	4695	SEC	
BEDE	E55E	4696	SBC Z5E	;compute complement of exponent
BEE0	AA	4697	TAX	
BEE1	A92D	4698	LDA ^-	;and set "-"
BEE3	990101	4699	BBEE3 STA X0101,Y	;store sign of exponent in output area
BEE6	A945	4700	LDA ^E	;set sign for exponential notation
BEE8	990001	4701	STA X0100,Y	;and store it
BEEB	8A	4702	TXA	
BEEC	A22F	4703	LDX \$2F	;get ASCII base for first exponent digit
BEEE	38	4704	SEC	
BEF0	E8	4705	BBEEF INX	;increment
BEF0	E90A	4706	SBC \$0A	;subtract 10 from exponent
BEF2	B0FB	4707	BCS BBEEF	;if not a borrow, repeat
BEF4	693A	4708	ADC \$3A	;convert second exponent digit to ASCII
BEF6	990301	4709	STA X0103,Y	;store second digit
BEF9	8A	4710	TXA	
BEFA	990201	4711	STA X0102,Y	;store first digit
BEFD	A900	4712	LDA \$00	
BEFF	990401	4713	STA X0104,Y	;store end of string
BFO2	F008	4714	BEQ BBFOC	;and exit

```

BF04 99FF00 4715 JBF04 STA X0100-1,Y ;store A at current output position
BF07 A900 4716 BBF07 LDA $00
BF09 990001 4717 STA X0100,Y ;store end of string
BFOC A900 4718 BBFOC LDA <X0100
BFOE A001 4719 LDY >X0100 ;set AY to point to string text
BF10 60 4720 RTS
      4721 ;
      4722 ;constants for conversions
      4723 ;
      4724 ;0.5 for rounding, SQR and dummy variable
BF11 800000 4725 TBF11 .BY $80,$00,$00,$00,$00
      4726 ;
      4727 ;divisors for decimal conversion
      4728 ;
      4729 ;-1000000000
BF16 FA0A1F 4730 TBF16 .BY $FA,$0A,$1F,$00
      4731 ;+100000000
BF1A 009896 4732 .BY $00,$98,$96,$80
      4733 ;-1000000
BF1E FFF0BD 4734 .BY $FF,$F0,$BD,$C0
      4735 ;+100000
BF22 000186 4736 .BY $00,$01,$86,$A0
      4737 ;-10000
BF26 FFFFD8 4738 .BY $FF,$FF,$D8,$F0
      4739 ;+1000
BF2A 000003 4740 .BY $00,$00,$03,$E8
      4741 ;-100
BF2E FFFFFFF 4742 .BY $FF,$FF,$FF,$9C
      4743 ;+10
BF32 000000 4744 .BY $00,$00,$00,$0A
      4745 ;-1
BF36 FFFFFFF 4746 .BY $FF,$FF,$FF,$FF
      4747 ;
      4748 ;divisors for time conversion
      4749 ;
      4750 ;-10 * 6 * 10 * 6 * 10 * 60
BF3A FFDFOA 4751 .BY $FF,$DF,$0A,$80
      4752 ;+6 * 10 * 6 * 10 * 60
BF3E 00034B 4753 .BY $00,$03,$4B,$C0
      4754 ;-10 * 6 * 10 * 60
BF42 FFFF73 4755 .BY $FF,$FF,$73,$60
      4756 ;+6 * 10 * 60
BF46 00000E 4757 .BY $00,$00,$0E,$10
      4758 ;-10 * 60
BF4A FFFF7D 4759 .BY $FF,$FF,$7D,$A8
      4760 ;+60
BF4E 000000 4761 .BY $00,$00,$00,$3C
      4762 ;
      4763 ;unused area follows
      4764 ;
BF52 ECAAAA 4765 .BY $EC,$AA,$AA,$AA,$AA,$AA,$AA,$AA
BF5A AAAAAA 4766 .BY $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
BF62 AAAAAA 4767 .BY $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
BF6A AAAAAA 4768 .BY $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA

```



```

4770 ;"SQR" command
4771 ;
BF71 200CBC 4772 WBF71 JSR SBCOC ;move rounded flp accu into 2nd flp accu
BF74 A911 4773 LDA <TBF11
BF76 A0BF 4774 LDY >TBF11 ;let AY point to value 0.5
BF78 20A2BB 4775 JSR SBBA2 ;load flp accu from AY
4776 ;
4777 ;"EXPONENT" operation
4778 ;
BF7B F070 4779 WBF7B BEQ BBFED ;if right operand zero, do EXP(0)
BF7D A569 4780 LDA Z69 ;if left operand zero
BF7F D003 4781 BNE BBF84
BF81 4CF9BB 4782 JMP JB8F9 ;set result zero
4783 ;
BF84 A24E 4784 BBF84 LDX <Z4E
BF86 A000 4785 LDY >Z4E ;set AY to Z4E
BF88 20D4BB 4786 JSR SBBD4 ;store flp accu there
BF8B A56E 4787 LDA Z6E ;get sign of left operand
BF8D 100F 4788 BPL BBF9E ;if negative
BF8F 20CCBC 4789 JSR SBCCC ;perform INT
BF92 A94E 4790 LDA <Z4E
BF94 A000 4791 LDY >Z4E ;set AY to Z4E again
BF96 205BBC 4792 JSR SBC5B ;compare AY with flp accu
BF99 D003 4793 BNE BBF9E ;if equal, right operand = integer
BF9B 98 4794 TYA ;set positive
BF9C A407 4795 LDY Z07 ;set sign change flag
BF9E 20FEBB 4796 BBF9E JSR SBBFE ;move second flp accu into flp accu with
BFA1 98 4797 TYA
BFA2 48 4798 PHA ;save sign change flag
BFA3 20EAB9 4799 JSR SB9EA ;perform LOG
BFA6 A94E 4800 LDA <Z4E
BFA8 A000 4801 LDY >Z4E ;set AY to Z4E
BFAA 2028BA 4802 JSR SBA28 ;multiply AY by flp accu
BFAD 20EDBF 4803 JSR BBFED ;perform EXP
BFBO 68 4804 PLA ;restore sign change flag
BFB1 4A 4805 LSR A ;shift into carry
BFB2 900A 4806 BCC BBFBE ;return if not set, else do monadic "-"
4807 ;
4808 ;"MINUS" operator
4809 ;
BFB4 4810 WBFB4 = *
BFB4 A561 4811 JBFB4 LDA Z61 ;if number = 0
BFB6 F006 4812 BEQ BBFBE ;so is result
BFB8 A566 4813 LDA Z66
BFB8 A9FF 4814 EOR $FF ;else invert sign
BFB8 8566 4815 STA Z66
BFB8 60 4816 BBFBE RTS

```

```

4818 ;floating point numbers for EXP
4819 ;
4820 ;1/LOG(2)
BFBF 8138AA 4821 TBFBF .BY $81,$38,$AA,$3B,$29
BFC4 07 4822 .BY $07 ;polynome table for EXP
4823 ;0.0000214987637
BFC5 713458 4824 .BY $71,$34,$58,$3E,$56
4825 ;0.000143523140
BFCA 74167E 4826 .BY $74,$16,$7E,$B3,$1B
4827 ;0.00134226348
BFCF 772FEE 4828 .BY $77,$2F,$EE,$E3,$85
4829 ;0.00961401701
BFD4 7A1D84 4830 .BY $7A,$1D,$84,$1C,$2A
4831 ;0.0555051269
BFD9 7C6359 4832 .BY $7C,$63,$59,$58,$0A
4833 ;0.240226385
BFDE 7E75FD 4834 .BY $7E,$75,$FD,$E7,$C6
4835 ;0.693147186
BFE3 803172 4836 .BY $80,$31,$72,$18,$10
4837 ;1.0
BFE8 810000 4838 .BY $81,$00,$00,$00,$00
4839 ;
4840 ;"EXP" command
4841 ;
BFED 4842 WBFED = *
BFED A9BF 4843 BBFED LDA <TBFBF
BFED A0BF 4844 LDY >TBFBF ;let AY point to 1/LOG(2)
BFF1 2028BA 4845 JSR SBA28 ;multiply AY times flp accu
BFF4 A570 4846 LDA Z70
BFF6 6950 4847 ADC $50 ;according to guard byte + 80
BFF8 9003 4848 BCC BBFFD
BFFA 2023BC 4849 JSR SBC23 ;round flp accu
BFFD 4C00E0 4850 BBFFD JMP XE000 ;continue in part 2

```

BA38F	A38F	503							
BA3A4	A3A4	489							
BA3B0	A3B0	495							
BA3B7	A3B7	487	498						
BA3DC	A3DC	535							
BA3E8	A3E8	547							
BA3EC	A3EC	541	543	553					
BA3F3	A3F3	530							
BA412	A412	574							
BA416	A416	583							
BA421	A421	589							
BA434	A434	573	576	594					
BA435	A435	563	567	595	597	2854			
BA456	A456	631							
BA474	A474	638							
BA480	A480	654	716	744					
BA49C	A49C	657							
BA4D7	A4D7	694							
BA4DF	A4DF	699	705	709					
BA4ED	A4ED	667							
BA508	A508	724							
BA522	A522	741							
BA53C	A53C	772							
BA544	A544	759							
BA55F	A55F	755							
BA562	A562	784							
BA576	A576	780							
BA582	A582	802	847						
BA58E	A58E	798							
BA5A4	A5A4	811							
BA5AC	A5AC	815							
BA5B6	A5B6	829							
BA5B8	A5B8	863							
BA5C7	A5C7	865							
BA5C9	A5C9	800	804	809	813	817	850	852	
BA5DC	A5DC	841							
BA5DE	A5DE	843							
BA5E5	A5E5	856							
BA5EE	A5EE	807							
BA5F5	A5F5	831							
BA5F9	A5F9	861							
BA609	A609	838							
BA617	A617	903	1298						
BA62E	A62E	890							
BA637	A637	892							
BA640	A640	884							
BA641	A641	889	896	897	909	980	983		
BA68D	A68D	927	974						
BA6A4	A6A4	971	972						
BA6BB	A6BB	978							
BA6C9	A6C9	988	1033						
BA6E6	A6E6	1007							
BA6E8	A6E8	1009							
BA6EF	A6EF	1061							
BA6F3	A6F3	1040	1042	1044					
BA700	A700	1018							
BA714	A714	998	1010	1023					
BA717	A717	1025							
BA72C	A72C	1055							

BA72F	A72F	1054	
BA737	A737	1051	1063
BA753	A753	1071	
BA79F	A79F	1111	
BA7BE	A7BE	1130	
BA7CE	A7CE	1139	
BA804	A804	1165	
BA807	A807	1135	
BA80B	A80B	1183	
BA80E	A80E	1167	
BA827	A827	1196	1923
BA82B	A82B	1163	
BA832	A832	1206	
BA849	A849	1216	
BA854	A854	1227	
BA862	A862	1236	
BA870	A870	1211	1233
BA87D	A87D	1254	
BA8BC	A8BC	1288	
BA8C0	A8C0	1293	1295
BA8D1	A8D1	1310	
BA8E3	A8E3	1299	
BA8E8	A8E8	1398	
BA8EB	A8EB	1316	
BA8FB	A8FB	1382	1800 1912
BA905	A905	1342	1360 1362
BA911	A911	1366	
BA919	A919	1365	
BA937	A937	1373	
BA940	A940	1377	
BA948	A948	1387	
BA953	A953	1426	
BA957	A957	1396	1406
BA95F	A95F	1400	
BA96A	A96A	1420	
BA99F	A99F	1442	
BA9D6	A9D6	1465	
BA9D9	A9D9	1463	
BA9ED	A9ED	1513	
BAA07	AA07	1506	
BAA24	AA24	1494	
BAA27	AA27	1525	
BAA2C	AA2C	1488	
BAA3D	AA3D	1537	
BAA4B	AA4B	1536	1541 1544
BAA52	AA52	1545	1548
BAA90	AA90	1589	
BAA9A	AA9A	1617	
BAA9D	AA9D	1622	
BAAE5	AAE5	1640	1698
BAAE7	AAE7	1605	1633 1692
BAAE8	AAE8	1612	
BAAEE	AAEE	1656	
BAAF8	AAF8	1607	1610
BAB0E	AB0E	1659	
BAB0F	AB0F	1668	
BAB10	AB10	1679	
BAB13	AB13	1614	1671
BAB19	AB19	1675	
BAB28	AB28	1697	1699

BAB42	AB42	1704
BAB57	AB57	1718
BAB5B	AB5B	1720
BAB5F	AB5F	1666
BAB62	AB62	1717
BAB6B	AB6B	1727
BAB92	AB92	1744
BABE7	ABE7	1758
BABD6	ABD6	1798
BABEA	ABEA	1788 1791
BAC03	AC03	1805
BAC0D	AC0D	1796
BAC41	AC41	1834
BAC4A	AC4A	1842
BAC4D	AC4D	1839
BAC51	AC51	1832 1917
BAC65	AC65	1851
BAC71	AC71	1856
BAC72	AC72	1859
BAC7D	AC7D	1868
BAC89	AC89	1849
BAC9D	AC9D	1879 1881
BACB8	ACB8	1840 1916
BACD1	ACD1	1900
BACDF	ACDF	1893
BACEA	ACEA	1922
BACFB	ACFB	1927 1929
BAD27	AD27	1946
BAD32	AD32	1904
BAD35	AD35	1951
BAD75	AD75	1994
BAD78	AD78	1977
BAD96	AD96	2016
BAD97	AD97	2012
BAD99	AD99	2013
BADA4	ADA4	2023
BADD7	ADD7	2039 2041
BADE8	ADE8	2058
BADF0	ADF0	2087
BADF9	ADF9	2090
BAE07	AE07	2053
BAE11	AE11	2082
BAE19	AE19	2074
BAE30	AE30	2047
BAE58	AE58	2054 2056
BAE5B	AE5B	2076
BAE5D	AE5D	2068
BAE64	AE64	2135
BAE66	AE66	2077 2089
BAE80	AE80	2133
BAE8A	AE8A	2188
BAE8F	AE8F	2184
BAE92	AE92	2165
BAE9A	AE9A	2169
BAEAD	AEAD	2173
BAEC6	AEC6	2194
BAECC	AEC6	2190
BAEE3	AE3	2200
BAEEA	AEEA	2217
BAEF1	AEF1	2221 2364 3008

BAF0D	AF0D	2186
BAF0F	AF0F	2202
BAF27	AF27	2256
BAF5C	AF5C	2275 2277 2279
BAF5D	AF5D	2271
BAF6E	AF6E	2293
BAF92	AF92	2306
BAFA0	AFA0	2304 2308 2327 2329
BAFD1	AFD1	2344
BB02E	B02E	2406
BB056	B056	2432 2434
BB05B	B05B	2449
BB066	B066	2442
BB072	B072	2444 2446 2451
BB07B	B07B	2457
BB07E	B07E	2467
BB09C	B09C	2501
BB09F	B09F	2478
BB0AF	B0AF	2485
BB0B0	B0B0	2490 2492
BB0BA	B0BA	2487
BB0C4	B0C4	2494
BB0D4	B0D4	2497
BB0DB	B0DB	2499
BB0E7	B0E7	2514
BB0EF	B0EF	2540
BB0F1	B0F1	2538
BB0FB	B0FB	2524
BB109	B109	2529
BB11C	B11C	2548
BB11D	B11D	2526
BB123	B123	2569
BB128	B128	2559
BB138	B138	2577
BB13E	B13E	2567 2571
BB143	B143	2575
BB159	B159	2588
BB185	B185	2533
BB18F	B18F	2622
BB1A0	B1A0	2635
BB1CC	B1CC	2658
BB1CE	B1CE	2664
BB1DB	B1DB	2704
BB21C	B21C	2737
BB228	B228	2718
BB237	B237	2725
BB245	B245	2750 2853
BB24A	B24A	2745
BB24D	B24D	2728
BB261	B261	2720
BB274	B274	2762
BB27D	B27D	2767
BB286	B286	2796
BB296	B296	2779
BB2B9	B2B9	2803
BB2C8	B2C8	2815 2818
BB2CD	B2CD	2812
BB2F2	B2F2	2870
BB308	B308	2848
BB30B	B30B	2798 2805 2911 2922

BA30E	B30E	2847
BB30F	B30F	2852
BB320	B320	2860
BB331	B331	2874
BB337	B337	2877
BB34B	B34B	2830 2960
BB35F	B35F	2924
BB378	B378	2914
BB384	B384	2930
BB3AE	B3AE	3020
BB418	B418	3026
BB449	B449	3048
BB497	B497	3111
BB4A4	B4A4	3109
BB4A8	B4A8	3107
BB4A9	B4A9	3113
BB4B5	B4B5	3120
BB4BF	B4BF	3124
BB4CA	B4CA	3126 3382 3480 3508
BB4D2	B4D2	3184
BB4D5	B4D5	3137
BB4F6	B4F6	3189
BB501	B501	3167
BB50B	B50B	3171
BB516	B516	3170 3173
BB544	B544	3211
BB54D	B54D	3209
BB559	B559	3223
BB561	B561	3219
BB566	B566	3221
BB56E	B56E	3252 3254
BB572	B572	3268
BB57D	B57D	3231 3233
BB5AE	B5AE	3262
BB5B0	B5B0	3270
BB5B8	B5B8	3266
BB5DC	B5DC	3294
BB5E6	B5E6	3300
BB5F6	B5F6	3276 3279 3287 3295 3297 3299 3302
BB601	B601	3315 3325
BB65D	B65D	3369
BB690	B690	3411
BB699	B699	3405
BB6A2	B6A2	3416
BB6D5	B6D5	3449
BB6D6	B6D6	3440 3442 3444
BB6EB	B6EB	3459 3461
BB70C	B70C	3487
BB70D	B70D	3535
BB70E	B70E	3538 3540
BB725	B725	3504
BB748	B748	3524
BB798	B798	3528 3579 3593 3642 3645
BB7B5	B7B5	3600
BB7CD	B7CD	3613
BB83C	B83C	3683
BB840	B840	3690
BB848	B848	3731
BB862	B862	3748
BB86F	B86F	3723

BB893	B893	3735		
BB897	B897	3744		
BB8A3	B8A3	3714	3734	
BB8AF	B8AF	3760		
BB8D7	B8D7	3778	4195	
BB8DB	B8DB	3796		
BB8FE	B8FE	3754		
BB91D	B91D	3828		
BB929	B929	3784		
BB946	B946	3835		
BB97D	B97D	3866	3871 3873 3875	
BB97E	B97E	3837	4080 4520	
BB985	B985	3896	3897	
BB9A6	B9A6	3912		
BB9AC	B9AC	3903		
BB9BA	B9BA	3901		
BB9F1	B9F1	3941		
BB9F4	B9F4	3942		
BBA30	BA30	3973		
BBA61	BA61	4023		
BBA7D	BA7D	4002		
BBAC4	BAC4	4059		
BBACF	BACF	4066		
BBADA	BADA	4056	4063	
BBADF	BADF	4060	4075 4089 4094 4125	
BBAF8	BAF8	4086		
BBB29	BB29	4154		
BBB3F	BB3F	4130	4133 4136 4153 4155	
BBB4C	BB4C	4141	4173	
BBB5D	BB5D	4148		
BBB7A	BB7A	4144		
BBB7E	BB7E	4145		
BBB8A	BB8A	4117		
BBC02	BC02	4270		
BBC11	BC11	4281		
BBC1A	BC1A	4288	4290 4292	
BBC2F	BC2F	4343		
BBC38	BC38	4298	4302	
BBC92	BC92	4345	4349 4353 4357	
BBC98	BC98	4365		
BBCAF	BCAF	4376		
BBCBA	BCBA	4363		
BBCBB	BCBB	4384		
BBCB9	BCE9	4372		
BBCF2	BCF2	4404		
BBCF7	BCF7	4432		
BBDO6	BD06	4435		
BBDOA	BD0A	4437	4469 4498	
BBDOD	BDOD	4433		
BBDOF	BD0F	4439		
BBD2E	BD2E	4449	4451	
BBD30	BD30	4453	4455 4533	
BBD33	BD33	4447		
BBD35	BD35	4456		
BBD41	BD41	4443		
BBD47	BD47	4445	4461	
BBD52	BD52	4478		
BBD5B	BD5B	4475	4482	
BBD62	BD62	4474	4479	
BBD67	BD67	4484		

BBD6A	BD6A	4441	
BBD71	BD71	4491	
BBD91	BD91	4459	
BDDA0	BDA0	4516	
BDDAE	BDAE	4519	
BBDE7	BDE7	4567	
BBDF8	BDF8	4575	
BBE00	BE00	4580	
BBE09	BE09	4581	
BBE0B	BE0B	4602	
BBE16	BE16	4599	
BBE21	BE21	4595	
BBE28	BE28	4591	
BBE2F	BE2F	4596	
BBE32	BE32	4590	
BBE47	BE47	4609	
BBE48	BE48	4611	
BBE53	BE53	4620	
BBE64	BE64	4627	
BBE66	BE66	4621	
BBE6A	BE6A	4649	4651 4681
BBE8E	BE8E	4648	
BBE90	BE90	4650	
BBE97	BE97	4653	
BBEB2	BEB2	4668	
BDEC4	BEC4	4679	
BDEC6	BEC6	4686	
BDED3	BED3	4688	
BBEE3	BEE3	4693	
BBEF	BEF	4707	
BBF07	BF07	4692	
BBF0C	BF0C	4714	
BBF84	BF84	4781	
BBF9E	BF9E	4788	4793
BBFBE	BFBE	4806	4812
BBFED	BFED	4779	4803
BBFFD	BFFD	4848	
JA437	A437	786	1237 1320 1729 1953 2018 2240 2742 2964 3139 3371 3880
		4185	
JA469	A469	1228	
JA7AE	A7AE	1159	1276 1986
JA7E1	A7E1	659	1151 1179
JA7EF	A7EF	1402	
JA84B	A84B	1140	
JA897	A897	1258	
JA8A0	A8A0	1187	1275 1388
JA8F8	A8F8	1793	2987
JA971	A971	1445	
JAA68	AA68	1551	
JAAA0	AAA0	1596	
JAAA2	AAA2	1677	
JAACA	AACA	788	
JAB4D	AB4D	1882	
JABB5	ABB5	1583	1792
JAC15	AC15	1895	
JAC91	AC91	1873	
JADA9	ADA9	2100	
JADB8	ADB8	3383	
JADBB	ADBB	2050	
JADFA	ADFA	2247	

JAE43	AE43	1104					
JAF08	AF08	1180	1322	1725	2102	2236	2479 2572 3049
JAF28	AF28	2170					
JAF7	AF7	2222					
JAFD6	AFD6	2362					
JB061	B061	2415					
JB1D1	B1D1	2515					
JB248	B248	1526	2668	3585	3943		
JB2EA	B2EA	2751					
JB391	B391	2212	2301	2401	2954		
JB3A2	B3A2	3568	3583	3667			
JB3F4	B3F4	2218					
JB44F	B44F	2988					
JB46F	B46F	2288					
JB52A	B52A	3351					
JB606	B606	3234					
JB63D	B63D	2059					
JB706	B706	3516					
JB86A	B86A	3711	4510				
JB8D2	B8D2	4325	4415				
JB8F7	B8F7	3601	3831	4078			
JB8F9	B8F9	4782					
JB8FB	B8FB	4067					
JB936	B936	3818					
JB938	B938	4293					
JB983	B983	3997					
JBA8B	BA8B	3974					
JBB12	BB12	4108					
JBB4F	BB4F	4170					
JBB8F	BB8F	3992	4182				
JBBDO	BBDO	1481	1971				
JBC31	BC31	4367					
JBC3C	BC3C	2331	2459	4505			
JBC44	BC44	2947					
JBC4F	BC4F	2312					
JBD49	BD49	4465					
JBD7E	BD7E	1529	3966	4497			
JBFO4	BF04	4576					
JBFB4	BFB4	4487					
SA38A	A38A	1070	1313	1950			
SA3B8	A3B8	727	2592				
SA3BF	A3BF	3341					
SA3FB	A3FB	1079	1263	2032			
SA408	A408	508	2756	2806			
SA533	A533	714	743				
SA560	A560	649	1808				
SA579	A579	658	664				
SA613	A613	666	976				
SA659	A659	713	742	1255			
SA660	A660	1257					
SA67A	A67A	632					
SA68E	A68E	922					
SA7ED	A7ED	1158	1389				
SA81D	A81D	939					
SA82C	A82C	999	1125				
SA906	A906	1080	1337	1799	1897		
SA909	A909	1282	1381				
SA96B	A96B	663	975	982	1281	1404	
SA9A5	A9A5	1069	1176				
SA9C2	A9C2	1877					

SA9DA	A9DA	1872							
SAA1D	AA1D	1499	1503						
SAA86	AA86	1582							
SAAD7	AAD7	622	1000	1604					
SAB1E	AB1E	635	642	1733	1932	4560			
SAB21	AB21	1598	1620	1782					
SAB3B	AB3B	1621	1678	1807					
SAB45	AB45	623	1806	1843					
SAR47	AB47	628	1016	1062	1638	1642	1694		
SABCE	ABCE	1768	1778						
SABF9	ABF9	1786	1844						
SACOF	ACOF	1756							
SAD24	AD24	1944	1996						
SAD8A	AD8A	1095	1113	3042	3590	3634			
SAD8D	AD8D	1094	2069	2136	2372	2656	2974	2999	3009 3074
SAD8F	AD8F	2348	3360	3422					
SAD90	AD90	1462	2405						
SAD9E	AD9E	1370	1459	1615	2000	2225	2346	2655	
SAE20	AE20	2071							
SAE33	AE33	2098							
SAE38	AE38	1115							
SAE83	AE83	2033	3359						
SAEBD	AEBD	1779							
SAEF7	AEF7	2706	2975	3544					
SAEFA	AEFA	2224	2345	2970					
SAEFD	AEFD	1894	2347	2463	3525	3636			
SAEFF	AEFF	1093	1186	1375	1454	1591	1748	1765	1781 2977 2993
SAF14	AF14	2274	2303						
SAF84	AF84	2280	2309						
SB08B	B08B	1450	1820	1947	2265	2973			
SB090	B090	2465							
SB092	B092	2996							
SB113	B113	2168	2477	2486	2491				
SB194	B194	2746	2755						
SB1B2	B1B2	2685							
SB1B8	B1B8	3591							
SB1BF	B1BF	1470	2206	2383	2391	2647			
SB34C	B34C	2791	2861						
SB355	B355	2882							
SB3A6	B3A6	1742	1783	2969					
SB3E1	B3E1	2968	3003						
SB475	B475	1555	3128	3373					
SB47D	B47D	3474	3494						
SB487	B487	1619	1683	2196	3081				
SB48D	B48D	1670							
SB4F4	B4F4	3089							
SB526	B526	584	2932	3185					
SB5BD	B5BD	3222							
SB5C7	B5C7	3210	3269						
SB67A	B67A	1560	3374						
SB688	B688	3131							
SB68C	B68C	3378	3507						
SB6A3	B6A3	3570							
SB6A6	B6A6	1492	1687	2420	2931				
SB6AA	B6AA	2426	3377	3381	3497				
SB6DB	B6DB	1568	3427						
SB761	B761	3484	3512	3527					
SB782	B782	3567	3578	3599					
SB79B	B79B	1664							
SB79E	B79E	1393	1588	1746	1763	2357	3526	3637	

SB7A1	B7A1	3470			
SB7E2	B7E2	1871	2197		
SB7EB	B7EB	3671	3679		
SB7F1	B7F1	3684			
SB7F7	B7F7	3635	3659		
SB849	B849	4603			
SB850	B850	3958			
SB867	B867	1970	3697	3952	3964
SB877	B877	4092			
SB947	B947	3779			
SB94D	B94D	4380			
SB96F	B96F	4291			
SB999	B999	3713	4385		
SB9B0	B9B0	3752	4395		
SB9EA	B9EA	4799			
SBA28	BA28	4584	4802	4845	
SEA59	BA59	3983	3985	3987	3989
SBA5E	BA5E	3991	3996		
SBABC	BA8C	3701	3718	3969	4112
SBAB7	BAB7	3976	4123		
SBAE2	BAE2	1500	1514	4480	4493 4597
SEAED	BAED	1509			
SBAFE	BAFE	4476	4600		
SBBOF	BBOF	3955			
SBBA2	BBA2	1108	1964	2176	2335 4107 4775
SBBD4	BBD4	3028	4230	4786	
SBBFC	BBFC	2390	3724		
SBFFE	BBFE	4796			
SBCOC	BCOC	1504	4084	4102	4503 4772
SBC1B	BC1B	1469	2116	4118	4239 4276
SBC23	BC23	4849			
SBC2B	BC2B	1114	3940	4308	4340
SBC49	BC49	4558			
SBC5B	BC5B	2413	2667	4589	4594 4792
SBC5D	BC5D	1973			
SBC9B	BC9B	1515	2669	3646	4405 4604
SBCCC	BCCC	4789			
SBCF3	BCF3	1875	2166	3622	
SBDC2	BDC2	639			
SBDCD	BDCD	1012			
SBDDA	BDDA	4548			
SBDDD	BDDD	1618			
SBDDF	BDDF	3076	4559		
SBE68	BE68	2287			
TA00C	A00C	1170	1172		
TA080	A080	2067	2088	2094	2096 2107
TA09E	A09E	828	860	862	1053 1060
TA19E	A19E	433			
TA1AC	A1AC	434			
TA1B5	A1B5	435			
TA1C2	A1C2	436			
TA1D0	A1D0	437			
TA1E2	A1E2	438			
TA1F0	A1F0	439			
TA1FF	A1FF	440			
TA210	A210	441			
TA225	A225	442			
TA235	A235	443			
TA23B	A23B	444			
TA24F	A24F	445			

TA25A	A25A	446	
TA26A	A26A	447	
TA272	A272	448	
TA27F	A27F	449	
TA290	A290	450	
TA29D	A29D	451	
TA2AA	A2AA	452	
TA2BA	A2BA	453	
TA2C8	A2C8	454	
TA2D5	A2D5	455	
TA2E4	A2E4	456	
TA2ED	A2ED	457	
TA300	A300	458	
TA30E	A30E	459	
TA31E	A31E	460	
TA324	A324	461	
TA328	A328	615	617
TA364	A364		
TA369	A369	633	634
TA371	A371	4546	4547
TA376	A376	640	641
TA381	A381	1225	1226
TA383	A383	462	
TACFC	ACFC	1930	1931
TADOC	ADOC	1731	1732
TAEA8	AEA8	2174	2175
TB1A5	B1A5	2665	2666
TB9BC	B9BC	1106	1107 3956 3957
TB9C1	B9C1	3959	3960
TB9D6	B9D6	3950	3951
TB9DB	B9DB	3953	3954
TB9E0	B9E0	3962	3963
TB9E5	B9E5	3967	3968
TBAF9	BAF9	4103	4104
TBDB3	BDB3	4592	4593
TBDB8	BDB8	4587	4588
TBDBD	BDBD	4582	4583
TBF11	BF11	3695	3696 4773 4774
TBF16	BF16	4636	4639 4642 4645
TBFBF	BFBF	4843	4844
W0314	O314		
W0316	O316		
W0318	O318		
W031A	O31A		
W031C	O31C		
W031E	O31E		
W0320	O320		
W0322	O322		
W0324	O324		
W0326	O326		
W0328	O328		
W032A	O32A		
W032C	O32C		
W032E	O32E		
W0330	O330		
W0332	O332		
WA642	A642	240	
WA65E	A65E	234	
WA69C	A69C	233	
WA742	A742	207	

WA78B	A78B	1100	1101
WA81D	A81D	218	
WA82F	A82F	222	
WA831	A831	206	
WA857	A857	232	
WA871	A871	216	
WA883	A883	219	
WA8A0	A8A0	215	
WA8D2	A8D2	220	
WA8F8	A8F8	209	
WA928	A928	217	
WA93B	A93B	221	
WA94B	A94B	223	
WA9A5	A9A5	214	
WAA80	AA80	230	
WAA86	AA86	235	
WAAA0	AAA0	231	
WAB7B	AB7B	239	
WABA5	ABA5	210	
WABBF	ABBF	211	
WAC06	AC06	213	
WAD1E	AD1E	208	
WAED4	AED4	296	
WAFE6	AFE6	290	
WAFE9	AFE9	287	
WBO16	BO16	299	
WB081	B081	212	
WB37D	B37D	248	
WB39E	B39E	249	
WB3B3	B3B3	228	
WB465	B465	260	
WB6EC	B6EC	263	
WB700	B700	264	
WB72C	B72C	265	
WB737	B737	266	
WB77C	B77C	259	
WB78B	B78B	262	
WB7AD	B7AD	261	
WB80D	B80D	258	
WB824	B824	229	
WB82D	B82D	224	
WB853	B853	275	
WB86A	B86A	272	
WB9EA	B9EA	252	
WBA2B	BA2B	278	
WBB12	BB12	281	
WBC39	BC39	244	
WBC58	BC58	246	
WBCCC	BCCC	245	
WBF71	BF71	250	
WBF7B	BF7B	284	
WBF84	BF84	293	
WBFED	BFED	253	
WEO97	EO97	251	
WE12A	E12A	236	
WE156	E156	226	
WE165	E165	227	
WE168	E168	225	
WE18E	E18E	237	
WE1C7	E1C7	238	

[illegible]

		2708	2929	2943	3151	3572								
Z0E	000E	1455	1876	2292	2483	2503	2674	2710						
Z0F	000F	796	808	844	996	1019	1021	1043	3161	3183	3187			
Z10	0010	955	1068	2500	2512	2518	2972	2995						
Z11	0011	1716	1817	1833	1850	1921								
Z12	0012	2140	2456											
Z13	0013	621	1593	1632	1639	1703	1726	1749	1757	1766	1769	1772	1787	
		1797	1804	1841	1928									
Z14	0014	728	893	986	990	1008	1285	1418	1427	1432	1433	1437	1439	
		1441	3649	3657	3661	3664	3674	3687						
Z15	0015	729	887	987	991	1006	1287	1419	1423	1435	1436	1438	1443	
		3650	3655	3666										
Z16	0016	944	3135	3156	3208	3462								
Z17	0017	3152	3460	3464										
Z18	0018	3458												
Z19	0019	943	3204	3205										
Z22	0022	523	533	539	564	566	616	625	675	698	702	750	754	
		758	762	765	769	770	1102	1424	1429	1431	1434	1523	1693	
		2062	2064	2110	2111	2127	2794	2866	2895	3136	3206	3216	3235	
		3238	3241	3244	3248	3256	3260	3261	3275	3278	3286	3289	3292	
		3305	3313	3314	3402	3408	3425	3430	3433	3436	3452	3502	3503	
		3581	3606	3609	4028	4031	4034	4037	4040	4048	4199	4202	4205	
		4208	4211	4216	4240	4244	4247	4250	4255	4258				
Z23	0023	618	673	700	706	751	766	771	1103	2113	3207	3217	3236	
		3263	3264	3306	3316	3317	3403	3426	3453	3505	3611	4029	4200	
		4241												
Z24	0024	684	703	1961	3610	3617	3620	3625	4334	4337	4341	4346	4351	
		4355	4361											
Z25	0025	677	696	707	3615	4335								
Z26	0026	3978	4013	4015	4016	4187								
Z27	0027	3979	4010	4012	4017	4189								
Z28	0028	2880	2897	2917	3980	4007	4009	4018	4191					
Z29	0029	2900	2920	3981	4004	4006	4019	4143	4193					
Z2B	002B	748	878	912	914	915	961	1193	1296					
Z2C	002C	749	879	919	964	1195	1297							
Z2D	002D	674	682	683	692	718	734	918	933	1547	2519	3214		
Z2E	002E	685	687	722	735	921	934	1543	2520	3215				
Z2F	002F	935	2525	2578	2596	2713	3220							
Z30	0030	936	2523	2579	2597	2714	3218							
Z31	0031	509	732	937	2582	2719	2807	2821	2935	3172	3200	3232		
Z32	0032	510	733	938	2583	2717	2808	2825	2938	3169	3201	3230		
Z33	0033	575	596	931	1540	2934	3165	3174	3195	3296	3337	3443	3447	
		3448												
Z34	0034	572	593	932	1535	2937	3166	3175	3196	3293	3338	3441	3450	
Z35	0035	3176	3409	3414	3415									
Z36	0036	3177	3417											

Z66	0066	1096	1497	1967	2106	2153	2437	2443	2657	3641	3705	3707	3738
		3799	3847	3849	4042	4070	4073	4212	4252	4265	4299	4324	4329
		4342	4364	4375	4390	4407	4408	4507	4566	4570	4813	4815	
Z67	0067	4436	4483										
Z68	0068	3893	4267	4279	4379	4386	4396						
Z69	0069	2142	2411	2412	3729	4049	4055	4780					
Z6A	006A	2144	2409	2410	3816	4014	4046	4128	4152	4166	4168		
Z6B	006B	2146	3813	4011	4038	4131	4151	4163	4165				
Z6C	006C	2148	2424	2427	2447	3810	4008	4035	4134	4150	4160	4162	
Z6D	006D	2150	2425	2428	3807	4005	4032	4137	4149	4157	4159		
Z6E	006E	2152	2407	3708	3737	4041	4044	4264	4506	4787			
Z6F	006F	1558	2154	3100	3106	3117	3129	3362	3366	3379	3389	3392	3395
		3709	3753	4043	4069	4091	4106	4508					
Z70	0070	1559	2273	3101	3119	3123	3130	3149	3364	3380	3726	3746	3750
		3765	3791	3793	3805	3823	3842	3862	3864	3865	3886	3900	3982
		4020	4180	4218	4259	4271	4282	4289	4323	4360	4406	4846	
Z71	0071	818	833	1498	1501	1502	1510	2283	2770	2792	2811	2838	2858
		2868	2912	3118	3604	3626	4571	4622	4631	4662	4672	4682	
Z72	0072	2758	2793	2810	2817	2839	2857	2871	2913	3122	3605	3627	
Z7A	007A	650	794	822	857	869	963	1083	1126	1134	1137	1143	1146
		1149	1150	1212	1240	1266	1291	1302	1330	1340	1341	1359	1736
		1823	1829	1845	1853	1865	1884	1890	1903	1906	1909	1983	2022
		2025	2081	2084	2191	2235	2985	3031	3034	3052	3602	3607	3628
		4529											
Z7B	007B	651	867	966	1085	1127	1152	1213	1241	1264	1292	1305	1332
		1343	1737	1824	1830	1846	1866	1885	1891	1985	2024	2083	2192
		2983	3029	3037	3054	3603	3612	3629					

```

1      .L
2      .H
3 ;CBM-64-Part Two

```

```

5 ;
0000 6 Z00 = $00 ;6510 data direction register
0001 7 Z01 = $01 ;6510 I/O register
      8 ; bit 0 (output) 0 = RAM at $A000-$BFFF (BASIC area)
      9 ; bit 1 (output) 0 = RAM at $E000-$FFFF (Kernal area)
     10 ; bit 2 (output) 0 = access CRT shapes at $D000-$DFFF
     11 ; bit 3 (output) cassette write line
     12 ; bit 4 (input) cassette sense line
     13 ; bit 5 (output) cassette motor control
     14 ; bit 6 unused
     15 ; bit 7 unused
0002 16 Z02 = $02 ;dummy address for offset
0003 17 Z03 = $03 ;fixed-float vector
0004 18 Z04 = $04 ;high byte of same
0005 19 Z05 = $05 ;fixed/float vector
0006 20 Z06 = $06 ;high byte of same
0007 21 Z07 = $07 ;search character
000A 22 Z0A = $0A ;0=load, 1=verify
0012 23 Z12 = $12 ;sign flag
0013 24 Z13 = $13 ;CMD file number
0016 25 Z16 = $16 ;pointer into temporary string stack
0018 26 Z18 = $18 ;high byte of last temp string vector
0022 27 Z22 = $22 ;utility pointer area
0023 28 Z23 = $23 ;
002B 29 Z2B = $2B ;pointer to start of BASIC
002C 30 Z2C = $2C ;high byte of same
002D 31 Z2D = $2D ;pointer start of variables
002E 32 Z2E = $2E ;high byte of same
0033 33 Z33 = $33 ;pointer to start of string storage
0034 34 Z34 = $34 ;high byte of same
0037 35 Z37 = $37 ;pointer to limit of memory
0038 36 Z38 = $38 ;high byte of same
0049 37 Z49 = $49 ;file/device save area
004A 38 Z4A = $4A ;second file/device save area
004E 39 Z4E = $4E ;misc. work area
0053 40 Z53 = $53 ;default step value
0054 41 Z54 = $54 ;JMP vector for functions
0056 42 Z56 = $56 ;
0057 43 Z57 = $57 ;misc. numeric work area
005C 44 Z5C = $5C ;
0061 45 Z61 = $61 ;floating point accu # 1 - exponent
0062 46 Z62 = $62 ;flp # 1 - mantissa
0063 47 Z63 = $63 ;
0064 48 Z64 = $64 ;
0065 49 Z65 = $65 ;
0066 50 Z66 = $66 ;flp # 1 - sign
0067 51 Z67 = $67 ;series evaluation constant pointer
0068 52 Z68 = $68 ;flp accu # 1 overflow
0069 53 Z69 = $69 ;flp accu # 2 thru Z6E
006E 54 Z6E = $6E
006F 55 Z6F = $6F ;sign comparison flp accu #1 vs accu # 2
0070 56 Z70 = $70 ;flp accu # 1 guard byte
0071 57 Z71 = $71 ;table pointer for EXP/SIN/ATN
0072 58 Z72 = $72 ;high byte of same
0073 59 Z73 = $73 ;character fetch code, beginning address

```

007A	60	Z7A	=	\$7A	;current character address
007B	61	Z7B	=	\$7B	;high byte of same
008B	62	Z8B	=	\$8B	;RND seed value
0090	63	Z90	=	\$90	;status word ST
0091	64	Z91	=	\$91	;keyboard scan results
0092	65	Z92	=	\$92	;tape timing speed correction
0093	66	Z93	=	\$93	;load=0, verify=1
0094	67	Z94	=	\$94	;serial output deferred character flag
0095	68	Z95	=	\$95	;serial deferred character
0096	69	Z96	=	\$96	;tape sync established flag
0097	70	Z97	=	\$97	;register save area
0098	71	Z98	=	\$98	;number of files open
0099	72	Z99	=	\$99	;input device
009A	73	Z9A	=	\$9A	;output device
009B	74	Z9B	=	\$9B	;tape character parity
009C	75	Z9C	=	\$9C	;tape byte available flag
009D	76	Z9D	=	\$9D	;direct/run mode (80/00)
009E	77	Z9E	=	\$9E	;tape pass 1 error log/char buffer
009F	78	Z9F	=	\$9F	;tape pass 2 error log
00A0	79	ZA0	=	\$A0	;jiffy clock - high byte
00A1	80	ZA1	=	\$A1	;jiffy clock - middle byte
00A2	81	ZA2	=	\$A2	;jiffy clock - low byte
00A3	82	ZA3	=	\$A3	;serial bit count/EOL flag
00A4	83	ZA4	=	\$A4	;serial byte read/tape cycle count
00A5	84	ZA5	=	\$A5	;serial bit count/tape hdr block count
00A6	85	ZA6	=	\$A6	;tape buffer pointer
00A7	86	ZA7	=	\$A7	;tape leader count/RS-232 input bit
00A8	87	ZA8	=	\$A8	;tape write new byte/read error/bit count
00A9	88	ZA9	=	\$A9	;tape write cycle/read error/start bit
00AA	89	ZAA	=	\$AA	;RS-232 input byte/tape scan/hdr count
00AB	90	ZAB	=	\$AB	;write leader length/read checksum/parity
00AC	91	ZAC	=	\$AC	;tape buffer/scrolling/I-O pointer
00AD	92	ZAD	=	\$AD	;high byte of same
00AE	93	ZAE	=	\$AE	;tape end address/end of I/O area
00AF	94	ZAF	=	\$AF	;high byte of same
00B0	95	ZB0	=	\$B0	;tape speed correction value
00B1	96	ZB1	=	\$B1	;work area for tape speed correction
00B2	97	ZB2	=	\$B2	;tape buffer pointer
00B3	98	ZB3	=	\$B3	;high byte of same
00B4	99	ZB4	=	\$B4	;RS-232 transmit bit count/tape sync
00B5	100	ZB5	=	\$B5	;RS-232 next bit to send/saved tape sync
00B6	101	ZB6	=	\$B6	;read char error/RS-232 output buffer
00B7	102	ZB7	=	\$B7	;number of characters in file name
00B8	103	ZB8	=	\$B8	;current logical file
00B9	104	ZB9	=	\$B9	;current secondary address
00BA	105	ZBA	=	\$BA	;current device
00BB	106	ZBB	=	\$BB	;pointer to file name
00BC	107	ZBC	=	\$BC	;high byte of same
00BD	108	ZBD	=	\$BD	;RS-232 xmit parity/tape read-write buffer
00BE	109	ZBE	=	\$BE	;tape phase
00BF	110	ZBF	=	\$BF	;tape input byte buffer
00C0	111	ZC0	=	\$C0	;tape motor interlock
00C1	112	ZC1	=	\$C1	;I/O start address/work address
00C2	113	ZC2	=	\$C2	;high byte of same
00C3	114	ZC3	=	\$C3	;kernel setup pointer
00C4	115	ZC4	=	\$C4	;high byte of same
00C5	116	ZC5	=	\$C5	;last key pressed
00C6	117	ZC6	=	\$C6	;keyboard buffer count
00C7	118	ZC7	=	\$C7	;reverse video switch
00C8	119	ZC8	=	\$C8	;end of line pointer for input

00C9	120 ZC9	=	\$C9	;input cursor, line number
00CA	121 ZCA	=	\$CA	;input cursor, position on line
00CB	122 ZCB	=	\$CB	;key pressed - \$40 if no key
00CC	123 ZCC	=	\$CC	;cursor enable - 0 = flash
00CD	124 ZCD	=	\$CD	;cursor flash timing countdown
00CE	125 ZCE	=	\$CE	;character under cursor
00CF	126 ZCF	=	\$CF	;cursor blink phase
00D0	127 ZD0	=	\$D0	;screen-kbd flag/screen line length
00D1	128 ZD1	=	\$D1	;pointer to screen line
00D2	129 ZD2	=	\$D2	;high byte of same
00D3	130 ZD3	=	\$D3	;position of cursor on line
00D4	131 ZD4	=	\$D4	;string flag
00D5	132 ZD5	=	\$D5	;screen line length current line
00D6	133 ZD6	=	\$D6	;cursor line number
00D7	134 ZD7	=	\$D7	;character-bit buffer/checksum
00D8	135 ZD8	=	\$D8	;# of outstanding inserts
00D9	136 ZD9	=	\$D9	;screen line table thru \$F0
00DA	137 ZDA	=	\$DA	; " " " "
00F1	138 ZF1	=	\$F1	;dummy screen link
00F3	139 ZF3	=	\$F3	;color memory pointer
00F4	140 ZF4	=	\$F4	;high byte of same
00F5	141 ZF5	=	\$F5	;keyboard table address
00F6	142 ZF6	=	\$F6	;high byte of same
00F7	143 ZF7	=	\$F7	;RS-232 receive buffer base address
00F8	144 ZF8	=	\$F8	;high byte of same
00F9	145 ZF9	=	\$F9	;RS-232 transmit buffer base address
00FA	146 ZFA	=	\$FA	;high byte of same

0014	148 X0014 =	\$0014	;integer value (SYS address)
0079	149 X0079 =	\$0079	;fetch current character
0100	150 X0100 =	\$0100	;flp to ASCII work area/tape error log
01FC	151 X01FC =	\$01FC	;prefix to line in input buffer
01FD	152 X01FD =	\$01FD	;high byte of same
0200	153 X0200 =	\$0200	;BASIC input buffer
0259	154 X0259 =	\$0259	;logical file table
0263	155 X0263 =	\$0263	;device # table
026D	156 X026D =	\$026D	;secondary address table
0277	157 X0277 =	\$0277	;keyboard buffer
0281	158 X0281 =	\$0281	;start of memory for operating system
0282	159 X0282 =	\$0282	;high byte of same
0283	160 X0283 =	\$0283	;top of memory for operating system
0284	161 X0284 =	\$0284	;high byte of same
0285	162 X0285 =	\$0285	;serial bus timeout flag (not used)
0286	163 X0286 =	\$0286	;current color code
0287	164 X0287 =	\$0287	;color under cursor
0288	165 X0288 =	\$0288	;screen memory page
0289	166 X0289 =	\$0289	;maximum size of keyboard buffer (10)
028A	167 X028A =	\$028A	;key repeat flag (\$8X=repeat all keys)
028B	168 X028B =	\$028B	;repeat key frequency counter
028C	169 X028C =	\$028C	;repeat key delay counter
028D	170 X028D =	\$028D	;keyboard shift/control flag
028E	171 X028E =	\$028E	;last keyboard shift pattern
028F	172 X028F =	\$028F	;address of keyboard decode routine
0290	173 X0290 =	\$0290	;high byte of same
0291	174 X0291 =	\$0291	;shift mode switch (\$80=locked)
0292	175 X0292 =	\$0292	;auto scroll flag (\$00=on)
0293	176 X0293 =	\$0293	;RS-232 Control Register
0294	177 X0294 =	\$0294	;RS-232 Command Register
0295	178 X0295 =	\$0295	;non-standard bit time
0296	179 X0296 =	\$0296	;high byte of same
0297	180 X0297 =	\$0297	;RS-232 Status Register
0298	181 X0298 =	\$0298	;number of bits to send/receive
0299	182 X0299 =	\$0299	;baud rate
029A	183 X029A =	\$029A	;high byte of same
029B	184 X029B =	\$029B	;RS-232 receive buffer input pointer
029C	185 X029C =	\$029C	;RS-232 receive buffer output pointer
029D	186 X029D =	\$029D	;RS-232 transmit buffer input pointer
029E	187 X029E =	\$029E	;RS-232 transmit buffer output pointer
029F	188 X029F =	\$029F	;IRQ vector save area
02A0	189 X02A0 =	\$02A0	;high byte of same
02A1	190 X02A1 =	\$02A1	;internal ICR2 activity register
02A2	191 X02A2 =	\$02A2	;internal CRB1 activity register
02A3	192 X02A3 =	\$02A3	;ICR1 save area
02A4	193 X02A4 =	\$02A4	;CRA1 save area
02A5	194 X02A5 =	\$02A5	;pos first 40 column line after cursor
02A6	195 X02A6 =	\$02A6	;US/Int1 machine flag (\$00 = US)
0300	196 X0300 =	\$0300	;error message link
030C	197 X030C =	\$030C	;save area for A register
030D	198 X030D =	\$030D	;save area for X register
030E	199 X030E =	\$030E	;save area for Y register
030F	200 X030F =	\$030F	;save area for SR (flag) register
0310	201 X0310 =	\$0310	;USR jump link
0311	202 X0311 =	\$0311	;low byte of USR jump address
0312	203 X0312 =	\$0312	;high byte of USR jump address
0314	204 X0314 =	\$0314	;IRQ vector
0315	205 X0315 =	\$0315	;high byte if IRQ vector
0316	206 X0316 =	\$0316	;BRK vector
0318	207 X0318 =	\$0318	;NMI vector

031A	208 X031A =	\$031A	;OPEN vector
031C	209 X031C =	\$031C	;CLOSE vector
031E	210 X031E =	\$031E	;set input vector
0320	211 X0320 =	\$0320	;set output vector
0322	212 X0322 =	\$0322	;restore I/O devices to default vector
0324	213 X0324 =	\$0324	;input vector
0326	214 X0326 =	\$0326	;output vector
0328	215 X0328 =	\$0328	;test STOP Key vector
032A	216 X032A =	\$032A	;GET vector
032C	217 X032C =	\$032C	;close all files and channels vector
0330	218 X0330 =	\$0330	;load RAM vector
0332	219 X0332 =	\$0332	;save RAM vector

8000	221	X8000 =	\$8000	;cartridge RESET vector
8002	222	X8002 =	\$8002	;cartridge RESTORE (warm start) vector
8004	223	X8004 =	\$8004	;cartridge identifier
A000	224	XA000 =	\$A000	;begin BASIC vector
A002	225	XA002 =	\$A002	;BASIC warm start vector
A364	226	TA364 =	\$A364	;message OK
A376	227	TA376 =	\$A376	;message READY
A408	228	XA408 =	\$A408	;array area overflow check
A437	229	XA437 =	\$A437	;error message link
A43A	230	XA43A =	\$A43A	;print error message
A474	231	XA474 =	\$A474	;print READY, do warm start
A483	232	WA483 =	\$A483	;standard BASIC warm start
A52A	233	XA52A =	\$A52A	;reset program ptrs and relink BASIC
A533	234	XA533 =	\$A533	;relink BASIC
A57C	235	WA57C =	\$A57C	;crunch tokens
A644	236	XA644 =	\$A644	;perform NEW
A663	237	XA663 =	\$A663	;perform CLR
A677	238	XA677 =	\$A677	;perform RESTORE, reset stack/pgm ptrs
A67A	239	XA67A =	\$A67A	;reset stack and program pointers
A68E	240	XA68E =	\$A68E	;re-initialize current character ptr
A71A	241	WA71A =	\$A71A	;print tokens
A7E4	242	WA7E4 =	\$A7E4	;execute a statement
AB1E	243	XAB1E =	\$AB1E	;print string from AY
AD8A	244	XAD8A =	\$AD8A	;get next non-string value
AD9E	245	XAD9E =	\$AD9E	;evaluate expression
AE86	246	WAE86 =	\$AE86	;fetch arithmetic element
AEFD	247	XAEFD =	\$AEFD	;check next character for ",",
AF08	248	XAF08 =	\$AF08	;print SYNTAX Error
B1AA	249	WB1AA =	\$B1AA	;flp-fixed routine
B248	250	WB248 =	\$B248	;print ILLEGAL QUANTITY Error
B391	251	WB391 =	\$B391	;fixed-flp routine
B6A3	252	XB6A3 =	\$B6A3	;de-allocate temporary string storage
B79E	253	XB79E =	\$B79E	;fetch integer value in X, check range
B7F7	254	XB7F7 =	\$B7F7	;convert flp to integer
B849	255	XB849 =	\$B849	;half round flp accu
B850	256	XB850 =	\$B850	;subtract flp accu from # indexed by AY
B853	257	XB853 =	\$B853	;perform diadic minus
B867	258	XB867 =	\$B867	;add flp accu to # indexed by AY
B8D7	259	XB8D7 =	\$B8D7	;perform postshift
B9BC	260	TB9BC =	\$B9BC	;flp literal 1
BA28	261	XBA28 =	\$BA28	;multiply flp accu times # indexed by AY
BAB9	262	XBAB9 =	\$BAB9	;add exponents
BAD4	263	XBAD4 =	\$BAD4	;check sign of flp accu
BB07	264	XBB07 =	\$BB07	;divide flp accu by # indexed by AY
BB0F	265	XBB0F =	\$BB0F	;divide # indexed by AY by flp accu
BBA2	266	XBBA2 =	\$BBA2	;load flp accu with # indexed by AY
BBC7	267	XBBC7 =	\$BBC7	;store flp accu at Z5C-60
BBCA	268	XBBCA =	\$BBCA	;store flp accu at Z57-5B
BBD4	269	XBBD4 =	\$BBD4	;store flp accu in area indexed by XY
BC0C	270	XBC0C =	\$BC0C	;move rounded flp accu into 2nd flp accu
BC0F	271	XBC0F =	\$BC0F	;move flp accu into 2nd flp accu
BC2B	272	XBC2B =	\$BC2B	;get sign of flp accu into A
BCCC	273	XBCCC =	\$BCCC	;perform INT
BD0C	274	XBD0C =	\$BD0C	;print # from AX
BFB4	275	XBFB4 =	\$BFB4	;minus operator
BFC4	276	TBFC4 =	\$BFC4	;polynome table for EXP


```

278 ;6567 video chip
279 ;
D000 280 XD000 = $D000 ;6567 video chip base address
D011 281 XD011 = $D011 ;bit 4 = 0 to disable video chip
D012 282 XD012 = $D012 ;Raster Register
D016 283 XD016 = $D016 ;bit 5 = 0 to reset video chip
D018 284 XD018 = $D018 ;memory pointers for video chip
D019 285 XD019 = $D019 ;Interrupt Register
D021 286 XD021 = $D021 ;Background # 0 color
287 ;
288 ;6581 Sound Interface Device
289 ;
D418 290 XD418 = $D418 ;mode/volume for SID chip
291 ;
292 ;6526 CIA1
293 ;
DC00 294 XDC00 = $DC00 ;PA1 port A
DC01 295 XDC01 = $DC01 ;PB1 port B
DC02 296 XDC02 = $DC02 ;DDRA1 direction bits for port A
DC03 297 XDC03 = $DC03 ;DDRB1 direction bits for port B
DC04 298 XDC04 = $DC04 ;TAL1 timer A low
DC05 299 XDC05 = $DC05 ;TAH1 timer A high
DC06 300 XDC06 = $DC06 ;TBL1 timer B low
DC07 301 XDC07 = $DC07 ;TBH1 timer B high
DC0D 302 XDC0D = $DC0D ;ICR1 interrupt control register
DC0E 303 XDC0E = $DC0E ;CRA1 control register for port A
DC0F 304 XDC0F = $DC0F ;CRB1 control register for port B
305 ;
306 ;6526 CIA2
307 ;
DD00 308 XDD00 = $DD00 ;PA2 port A
DD01 309 XDD01 = $DD01 ;PB2 port B
DD02 310 XDD02 = $DD02 ;DDRA2 direction bits for port A
DD03 311 XDD03 = $DD03 ;DDRB2 direction bits for port B
DD04 312 XDD04 = $DD04 ;TAL2 timer A low
DD05 313 XDD05 = $DD05 ;TAH2 timer A high
DD06 314 XDD06 = $DD06 ;TBL2 timer B low
DD07 315 XDD07 = $DD07 ;TBH2 timer B high
DD0D 316 XDD0D = $DD0D ;ICR2 interrupt control register
DD0E 317 XDD0E = $DD0E ;CRA2 control register for port A
DD0F 318 XDD0F = $DD0F ;CRB2 control register for port B

```

```

320 ;continuation of "EXP" routine
321 ;
E000 322 .OR $E000
E000 8556 323 STA Z56 ;set guard byte for second flp accu
E002 200FBC 324 JSR XBC0F ;copy flp accu into second flp accu
E005 A561 325 LDA Z61 ;get exponent
E007 C988 326 CMP $88 ;if => 8
E009 9003 327 BCC BE00E
E00B 20D4BA 328 BE00B JSR XBAD4 ;then either 0 or overflow
E00E 20CCBC 329 BE00E JSR XBCCC ;perform INT
E011 A507 330 LDA Z07 ;get least significant byte
E013 18 331 CLC
E014 6981 332 ADC $81 ;if = 127
E016 F0F3 333 BEQ BE00B ;then also overflow
E018 38 334 SEC
E019 E901 335 SBC $01 ;remove excess 128
E01B 48 336 PHA ;and save it
E01C A205 337 LDX $05
E01E B569 338 BE01E LDA Z69,X ;move second flp accu into flp accu
E020 B461 339 LDY Z61,X
E022 9561 340 STA Z61,X ;and vice versa
E024 9469 341 STY Z69,X
E026 CA 342 DEX
E027 10F5 343 BPL BE01E
E029 A556 344 LDA Z56
E02B 8570 345 STA Z70 ;restore copied guard byte
E02D 2053B8 346 JSR XB853 ;apply diadic "-"
E030 20B4BF 347 JSR XBF84 ;then minus operator
E033 A9C4 348 LDA <TBFC4
E035 A0BF 349 LDY >TBFC4 ;set AY to polynome table
E037 2059E0 350 JSR SE059 ;and compute polynome
E03A A900 351 LDA $00
E03C 856F 352 STA Z6F ;clear XOR of signs
E03E 68 353 PLA ;add saved integral part
E03F 20B9BA 354 JSR XBAB9 ;to exponent
E042 60 355 RTS

```

```

357 ;compute odd degrees for SIN and ATN
358 ;
E043 8571 359 JE043 STA Z71 ;save table pointer
E045 8472 360 STY Z72
E047 20CARB 361 JSR XBBCA ;store flp accu at Z57
E04A A957 362 LDA <Z57 ;set AY to Z57
E04C 2028BA 363 JSR XBA28 ;multiply flp accu by # indexed by AY
E04F 205DE0 364 JSR SE05D ;compute polynome
E052 A957 365 LDA <Z57 ;set AY to Z57
E054 A000 366 LDY >Z57
E056 4028BA 367 JMP XBA28 ;multiply flp accu by Z57
368 ;
369 ;compute polynome according to table indexed by AY
370 ;
E059 8571 371 SE059 STA Z71 ;save table pointer
E05B 8472 372 STY Z72
E05D 20C7BB 373 SE05D JSR XBBC7 ;store flp accu at Z5C
E060 B171 374 LDA (Z71),Y
E062 8567 375 STA Z67 ;save order
E064 A471 376 LDY Z71
E066 C8 377 INY ;add 1 to table pointer
E067 98 378 TYA
E068 D002 379 BNE BE06C
E06A E672 380 INC Z72
E06C 8571 381 BE06C STA Z71 ;save updated pointer
E06E A472 382 LDY Z72 ;and restore AY
E070 2028BA 383 BE070 JSR XBA28 ;multiply flp accu by # indexed by AY
E073 A571 384 LDA Z71 ;restore AY
E075 A472 385 LDY Z72
E077 18 386 CLC
E078 6905 387 ADC $05 ;add 5 to table pointer
E07A 9001 388 BCC BE07D
E07C C8 389 INY
E07D 8571 390 BE07D STA Z71 ;save table pointer
E07F 8472 391 STY Z72
E081 2067B8 392 JSR XB867 ;add # indexed by AY to flp accu
E084 A95C 393 LDA <Z5C
E086 A000 394 LDY >Z5C ;set AY to Z5C
E088 C667 395 DEC Z67 ;decrement order
E08A D0E4 396 BNE BE070 ;and repeat until 0
E08C 60 397 RTS

```

```

399 ;floating point numbers for RND
400 ;
401 ; flp number for multiplication
E08D 983544 402 TE08D .BY $98,$35,$44,$7A,$00
403 ; flp number for addition
E092 6828B1 404 TE092 .BY $68,$28,$B1,$46,$00
405 ;
406 ;"RND" command
407 ;
E097 202BBC 408 WE097 JSR XBC2B ;get sign of flp accu into A
E09A 3037 409 BMI BE0D3 ;if < 0 use parameter as seed
E09C D020 410 BNE BE0BE ;if > 0 use old seed
E09E 20F3FF 411 JSR SFFF3 ;get base address of I/O devices
EOA1 8622 412 STX Z22
EOA3 8423 413 STY Z23
EOA5 A004 414 LDY $04
EOA7 B122 415 LDA (Z22),Y ;TAL1 timer value
EOA9 8562 416 STA Z62
EOAB C8 417 INY
EOAC B122 418 LDA (Z22),Y ;TAH2 timer vaue
EOAE 8564 419 STA Z64
EOBO A008 420 LDY $08
EOB2 B122 421 LDA (Z22),Y ;tenths of a second
EOB4 8563 422 STA Z63
EOB6 C8 423 INY
EOB7 B122 424 LDA (Z22),Y ;seconds
EOB9 8565 425 STA Z65
EOBB 4CE3E0 426 JMP JE0E3 ;garble flp set from CIA1 timer/clock
427 ;
EOBE A98B 428 BE0BE LDA <Z8B ;let AY point to Z8B (current seed)
EOC0 A000 429 LDY >Z8B
EOC2 20A2BB 430 JSR XBBA2 ;load flp accu from Z8B-Z8F
EOC5 A98D 431 LDA <TE08D ;set AY to RND factor
EOC7 A0E0 432 LDY >TE08D
EOC9 2028BA 433 JSR XBA28 ;multiply flp accu times RND factor
EOCC A992 434 LDA <TE092 ;set AY to RND addition value
EOCE A0E0 435 LDY >TE092
EOD0 2067B8 436 JSR XB867 ;add RND factor to result in flp accu
EOD3 A665 437 BE0D3 LDX Z65 ;exchange byte 4 and byte 1
EOD5 A562 438 LDA Z62
EOD7 8565 439 STA Z65
EOD9 8662 440 STX Z62
EODB A663 441 LDX Z63 ;exchange byte 2 and byte 3
EODD A564 442 LDA Z64
EODF 8563 443 STA Z63
EOE1 8664 444 STX Z64
EOE3 A900 445 JE0E3 LDA $00 ;set sign positive
EOE5 8566 446 STA Z66
EOE7 A561 447 LDA Z61
EOE9 8570 448 STA Z70 ;set guard byte
EOEB A980 449 LDA $80 ;move 0 to exponent
EOED 8561 450 STA Z61
EOEF 20D7B8 451 JSR XB8D7 ;perform postshift
EOF2 A28B 452 LDX <Z8B ;set XY to RND seed value
EOF4 A000 453 LDY >Z8B
EOF6 4CD4BB 454 SE0F6 JMP XBBD4 ;store flp accu as new seed

```

```

                                456 ;handle errors for direct I/O calls from BASIC interpreter
                                457 ;
EOF9 C9F0 458 BEOF9 CMP $F0      ;if open for RS-232,
EOF8 D007 459      BNE BE104
EOF8 8438 460      STY Z38      ;set new top of memory limit
EOF8 8637 461      STX Z37
E101 4C63A6 462      JMP XA663  ;CLR, back to basic
                                463 ;
E104 AA 464 BE104 TAX
E105 D002 465      BNE BE109    ;if no error message indexed,
E107 A21E 466      LDX $1E      ;select BREAK message
E109 4C37A4 467 BE109 JMP XA437  ;go print message
                                468 ;
E10C 20D2FF 469      JSR SFFD2   ;output a character
E10F B0E8 470      BCS BEOF9    ;if no errors,
E111 60 471      RTS           ;return
                                472 ;
E112 20CFFF 473      JSR SFFCF   ;input a character on current device
E115 B0E2 474      BCS BEOF9    ;if no errors,
E117 60 475      RTS           ;return
                                476 ;
E118 20ADE4 477      JSR SE4AD   ;set output device
E11B B0DC 478      BCS BEOF9    ;if no errors,
E11D 60 479      RTS           ;return
                                480 ;
E11E 20C6FF 481      JSR SFFC6   ;set input device
E121 B0D6 482      BCS BEOF9    ;if no errors,
E123 60 483      RTS           ;return
                                484 ;
E124 20E4FF 485      JSR SFFE4   ;get a character from current device
E127 B0DC 486      BCS BEOF9    ;if no errors,
E129 60 487      RTS           ;return

```

```

489 ;"SYS" command
490 ;
E12A 208AAD 491 WE12A JSR XAD8A ;get next non-string value
E12D 20F7B7 492 JSR XB7F7 ;convert to integer in Z14/15
E130 A9E1 493 LDA >WE147-1
E132 48 494 PHA
E133 A946 495 LDA <WE147-1
E135 48 496 PHA ;set return address
E136 AD0F03 497 LDA X030F
E139 48 498 PHA ;save flag register on stack
E13A AD0C03 499 LDA X030C
E13D AE0D03 500 LDX X030D ;restore A, X & Y
E140 AC0E03 501 LDY X030E ;to status of exit from last SYS
E143 28 502 PLP
E144 6C1400 503 JMP (X0014) ;do SYS routine
504 ;
E147 08 505 WE147 PHP ;save flag register on stack
E148 8D0C03 506 STA X030C ;save A
E14B 8E0D03 507 STX X030D ;save X
E14E 8C0E03 508 STY X030E ;save Y
E151 68 509 PLA
E152 8D0F03 510 STA X030F ;save Flag register
E155 60 511 RTS

```

```

513 ;"SAVE" command
514 ;
E156 20D4E1 515 WE156 JSR SE1D4 ;set file parameters
E159 A62D 516 LDX Z2D ;set end address of BASIC text
E15B A42E 517 LDY Z2E
E15D A92B 518 LDA <Z2B ;set address of start address
E15F 20D8FF 519 JSR SFFD8 ;save RAM to a device
E162 B095 520 BCS BE0F9 ;if no errors,
E164 60 521 RTS ;return
522 ;
523 ;"VERIFY" command
524 ;
E165 A901 525 WE165 LDA $01 ;set value for Verify
E167 2C 526 .BY $2C ;skip next instruction
527 ;
528 ;"LOAD" command
529 ;
E168 A900 530 WE168 LDA $00 ;set value for Load
E16A 850A 531 STA Z0A ;set load/verify flag
E16C 20D4E1 532 JSR SE1D4 ;fetch file parameters
E16F A50A 533 LDA Z0A ;fetch load/verify flag
E171 A62B 534 LDX Z2B ;set start address in XY
E173 A42C 535 LDY Z2C
E175 20D5FF 536 JSR SFFD5 ;load device to RAM
E178 B057 537 BCS BE1D1 ;exit upon error
E17A A50A 538 LDA Z0A
E17C F017 539 BEQ BE195 ;if verify pass,
E17E A21C 540 LDX $1C ;index Verify Error message
E180 20B7FF 541 JSR SFFB7 ;read ST
E183 2910 542 AND $10 ;if ST indicates a mismatch,
E185 D017 543 BNE BE19E ;go print error
E187 A57A 544 LDA Z7A
E189 C902 545 CMP $02
E18B F007 546 BEQ BE194 ;exit if in RUN mode, else
E18D A964 547 LDA <TA364 ;set AY to point to message OK
E18F A0A3 548 LDY >TA364
E191 4C1EAB 549 JMP XAB1E ;print message OK
E194 60 550 BE194 RTS
551 ;
E195 20B7FF 552 BE195 JSR SFFB7 ;if load pass
E198 29BF 553 AND $BF ;and ST indicates an error
E19A F005 554 BEQ BE1A1
E19C A21D 555 LDX $1D ;index Load Error message
E19E 4C37A4 556 BE19E JMP XA437 ;print message
557 ;
E1A1 A57B 558 BE1A1 LDA Z7B ;check high addr of current character
E1A3 C902 559 CMP $02
E1A5 D00E 560 BNE BE1E5 ;if in direct mode
E1A7 862D 561 STX Z2D ;move end of program address
E1A9 842E 562 STY Z2E ;to end of BASIC address
E1AB A976 563 LDA <TA376 ;set AY to READY message
E1AD A0A3 564 LDY >TA376
E1AF 201EAB 565 JSR XAB1E ;print READY
E1B2 4C2AA5 566 JMP XA52A ;relink BASIC, exit
567 ;
568 ;end of load/verify from within a program
569 ;
E1B5 208EA6 570 BE1B5 JSR XA68E ;move start of program to next char addr
E1B8 2033A5 571 JSR XA533 ;relink BASIC
E1BB 4C77A6 572 JMP XA677 ;restore, clear pointers, return

```

```

574 ;"OPEN" command
575 ;
E1BE 2019E2 576 WE1BE JSR SE219 ;fetch parameters
E1C1 20C0FF 577 JSR SFFC0 ;perform open
E1C4 B00B 578 BCS BE1D1 ;if no errors,
E1C6 60 579 RTS ;return
580 ;
581 ;"CLOSE" command
582 ;
E1C7 2019E2 583 WE1C7 JSR SE219 ;fetch parameters
E1CA A549 584 LDA Z49 ;restore file #
E1CC 20C3FF 585 JSR SFFC3 ;perform close
E1CF 90C3 586 BCC BE194 ;exit if no errors
E1D1 4CF9E0 587 BE1D1 JMP BEOF9 ;error
588 ;
589 ;set parameters for load/verify/save
590 ;
E1D4 A900 591 SE1D4 LDA $00 ;assume no file name
E1D6 20BDFF 592 JSR SFFBD ;set file name parameters
E1D9 A201 593 LDX $01 ;assume device 1
E1DB A000 594 LDY $00 ;and secondary address 0
E1DD 20BAFF 595 JSR SFFBA ;set logical file, device & sec. addr
E1E0 2006E2 596 JSR SE206 ;fetch current character
E1E3 2057E2 597 JSR SE257 ;evaluate expression
E1E6 2006E2 598 JSR SE206 ;fetch current character
E1E9 2000E2 599 JSR SE200 ;skip comma and fetch integer into X
E1EC A000 600 LDY $00 ;secondary address still 0
E1EE 8649 601 STX Z49 ;save device #
E1F0 20BAFF 602 JSR SFFBA ;set logical file, device & sec. addr
E1F3 2006E2 603 JSR SE206 ;fetch current character
E1F6 2000E2 604 JSR SE200 ;skip comma and fetch integer into X
E1F9 8A 605 TXA
E1FA A8 606 TAY ;set secondary address
E1FB A649 607 LDX Z49 ;restore device
E1FD 4CBAFF 608 JMP SFFBA ;set logical file, device & sec. addr
609 ;
610 ;skip comma and fetch integer into X
611 ;
E200 200EE2 612 SE200 JSR SE20E ;check for and skip comma
E203 4C9EB7 613 JMP XB79E ;fetch integer value into X
614 ;
615 ;fetch current character and check for end of line
616 ;
E206 207900 617 SE206 JSR X0079 ;fetch current character
E209 D002 618 BNE BE20D
E20B 68 619 PLA ;if end of line, delete own return addr
E20C 68 620 PLA
E20D 60 621 BE20D RTS
622 ;
623 ;check for a comma and skip it
624 ;
E20E 20FDAE 625 SE20E JSR XAEFD ;check for and skip ",",
E211 207900 626 SE211 JSR X0079 ;fetch current character
E214 D0F7 627 BNE BE20D
E216 4C08AF 628 JMP XAF08 ;SYNTAX Error if end of line reached

```



```

630 ;get Open/Close parameters
631 ;
E219 A900 632 SE219 LDA $00 ;assume no file name
E21B 20BDF 633 JSR SFFBD ;set file name parameters
E21E 2011E2 634 JSR SE211 ;fetch current character
E221 209EB7 635 JSR XB79E ;get next non-string value into X
E224 8649 636 STX Z49 ;save file number
E226 8A 637 TXA
E227 A201 638 LDX $01 ;assume device 1
E229 A000 639 LDY $00 ;and secondary address 0
E22B 20BAFF 640 JSR SFFBA ;set currrent file, device & sec. addr
E22E 2006E2 641 JSR SE206 ;fetch current character
E231 2000E2 642 JSR SE200 ;skip comma and fetch integer value into X
E234 864A 643 STX Z4A ;save device number
E236 A000 644 LDY $00 ;secondary still 0
E238 A549 645 LDA Z49
E23A E003 646 CFX $03
E23C 9001 647 BCC BE23F ;if device number > 3
E23E 88 648 DEY ;secondary = $FF
E23F 20BAFF 649 BE23F JSR SFFBA ;set logical file, device & sec. addr
E242 2006E2 650 JSR SE206 ;fetch current character
E245 2000E2 651 JSR SE200 ;skip comma,}fetch integer value in X
E248 8A 652 TXA
E249 A8 653 TAY ;set secondary address
E24A A64A 654 LDX Z4A ;restore device
E24C A549 655 LDA Z49 ;and file
E24E 20BAFF 656 JSR SFFBA ;set logical file, device & sec. addr
E251 2006E2 657 JSR SE206 ;fetch current character
E254 200EE2 658 JSR SE20E ;check for and skip ", "
E257 209EAD 659 SE257 JSR XAD9E ;evaluate expression
E25A 20A3B6 660 JSR XB6A3 ;de-allocate temporary string
E25D A622 661 LDX Z22
E25F A423 662 LDY Z23 ;XY = address of file name
E261 4CBDF 663 JMP SFFBD ;set file name parameters

```

```

        665 ;"COS" command
        666 ;
E264 A9E0 667 WE264 LDA <TE2E0 ;set AY to point to 0.5 * PI
E266 A0E2 668 LDY >TE2E0
E268 2067B8 669 JSR XB867 ;add to flp accu then do SIN
        670 ;
        671 ;"SIN" command
        672 ;
E26B 200CBC 673 SE26B JSR XBCOC ;move rounded flp accu to 2nd flp accu
E26E A9E5 674 LDA <TE2E5 ;set AY to 2 * PI
E270 A0E2 675 LDY >TE2E5
E272 A66E 676 LDX Z6E ;get sign
E274 2007BB 677 JSR XBB07 ;divide flp accu by 2 * PI
E277 200CBC 678 JSR XBCOC ;move rounded flp accu to 2nd flp accu
E27A 20CCBC 679 JSR XBCCC ;call function INT
E27D A900 680 LDA $00
E27F 856F 681 STA Z6F ;set XOR of signs to be the same
E281 2053B8 682 JSR XB853 ;apply diadic operator "-"
E284 A9EA 683 LDA <TE2EA ;set AY to point to 0.25
E286 A0E2 684 LDY >TE2EA
E288 2050B8 685 JSR XB850 ;subtract flp accu from 0.25
E28B A566 686 LDA Z66 ;flp accu # 1 - sign
E28D 48 687 PHA ;save sign
E28E 100D 688 BPL BE29D ;if negative
E290 2049B8 689 JSR XB849 ;do half rounding (add 0.5)
E293 A566 690 LDA Z66 ;get sign
E295 3009 691 BMI BE2A0
E297 A512 692 LDA Z12 ;if positive, complement flag
E299 49FF 693 EOR $FF
E29B 8512 694 STA Z12
E29D 20B4BF 695 BE29D JSR XBF84 ;apply monadic operator "-"
E2A0 A9EA 696 BE2A0 LDA <TE2EA ;set AY to 0.25
E2A2 A0E2 697 LDY >TE2EA
E2A4 2067B8 698 JSR XB867 ;add 0.25 to flp accu
E2A7 68 699 PLA ;restore original sign
E2A8 1003 700 BPL BE2AD ;if negative,
E2AA 20B4BF 701 JSR XBF84 ;perform monadic "-" again
E2AD A9EF 702 BE2AD LDA <TE2EF ;set AY to polynome table
E2AF A0E2 703 LDY >TE2EF
E2B1 4C43C0 704 JMP JE043 ;go compute odd degrees

```

```

706 ; "TAN" command
707 ;
E2B4 20CABB 708 WE2B4 JSR XBBCA ;store flp accu at Z57-5B
E2B7 A900 709 LDA $00
E2B9 8512 710 STA Z12 ;clear sign flag
E2BB 206BE2 711 JSR SE26B ;perform SIN
E2BE A24E 712 LDX <Z4E ;set AY to Z4E
E2C0 A000 713 LDY >Z4E
E2C2 20F6E0 714 JSR SE0F6 ;store flp accu at Z4E
E2C5 A957 715 LDA <Z57 ;set AY to Z57
E2C7 A000 716 LDY >Z57
E2C9 20A2BB 717 JSR XBBA2 ;move flp accu there
E2CC A900 718 LDA $00
E2CE 8566 719 STA Z66 ;clear sign of flp accu
E2D0 A512 720 LDA Z12 ;fetch sign flag
E2D2 20DCE2 721 JSR SE2DC ;compute COS of flp accu
E2D5 A94E 722 LDA <Z4E ;set AY to Z4E
E2D7 A000 723 LDY >Z4E
E2D9 4C0FBB 724 JMP XBB0F ;divide number at Z4E by flp accu, exit
725 ;
E2DC 48 726 SE2DC PHA
E2DD 4C9DE2 727 JMP BE29D ;compute COS of flp accu then return
728 ;
729 ;flp numbers for SIN, COS and TAN
730 ;
731 ;0.5 * PI
E2E0 81490F 732 TE2E0 .BY $81,$49,$0F,$DA,$A2
733 ;2 * PI
E2E5 83490F 734 TE2E5 .BY $83,$49,$0F,$DA,$A2
735 ;0.25
E2EA 7F0000 736 TE2EA .BY $7F,$00,$00,$00,$00
737 ;
E2EF 05 738 TE2EF .BY $05 ;polynome table for SIN
739 ;-14.3813907
E2F0 84E61A 740 .BY $84,$E6,$1A,$2D,$1B
741 ; 42.0077971
E2F5 862807 742 .BY $86,$28,$07,$FB,$F8
743 ;-76.7041703
E2FA 879968 744 .BY $87,$99,$68,$89,$01
745 ; 81.6052237
E2FF 872335 746 .BY $87,$23,$35,$DF,$E1
747 ;-41.3417021
E304 86A55D 748 .BY $86,$A5,$5D,$E7,$28
749 ; 6.28318531
E309 83490F 750 .BY $83,$49,$0F,$DA,$A2

```

```

752 ;"ATN" command
753 ;
E30E A566 754 WE30E LDA Z66 ;save sign
E310 48 755 PHA
E311 1003 756 BPL BE316 ;if negative,
E313 20B4BF 757 JSR XBFB4 ;apply monadic operator "-"
E316 A561 758 BE316 LDA Z61
E318 48 759 PHA ;save exponent
E319 C981 760 CMP $81 ;if => 1
E31B 9007 761 BCC BE324
E31D A9BC 762 LDA <TB9BC ;let AY point to 1
E31F A0B9 763 LDY >TB9BC
E321 200FBB 764 JSR XBB0F ;divide 1 by flp accu
E324 A93E 765 BE324 LDA <TE33E ;set AY to polynome table for ATN
E326 A0E3 766 LDY >TE33E
E328 2043E0 767 JSR JE043 ;compute odd degrees
E32B 68 768 PLA
E32C C981 769 CMP $81 ;if exponent => 1
E32E 9007 770 BCC BE337
E330 A9E0 771 LDA <TE2E0 ;set AY to 0.5 * PI
E332 A0E2 772 LDY >TE2E0
E334 2050B8 773 JSR XBB50 ;subtract flp accu from 0.5 * PI
E337 68 774 BE337 PLA
E338 1003 775 BPL BE33D ;if sign negative
E33A 4CB4BF 776 JMP XBFB4 ;apply monadic operator "-"
777 ;
E33D 60 778 BE33D RTS
779 ;
780 ;flp numbers for ATN
781 ;
E33E 0B 782 TE33E .BY $0B
783 ;-0.000684793912
E33F 76B383 784 .BY $76,$B3,$83,$BD,$D3
785 ; 0.00485094216
E344 791EF4 786 .BY $79,$1E,$F4,$A6,$F5
787 ;-0.0131117018
E349 7B83FC 788 .BY $7B,$83,$FC,$B0,$10
789 ; 0.034209638
E34E 7C0C1F 790 .BY $7C,$0C,$1F,$67,$CA
791 ;-0.0542791328
E353 7CDE53 792 .BY $7C,$DE,$53,$CB,$C1
793 ; 0.0724571965
E358 7D1464 794 .BY $7D,$14,$64,$70,$4C
795 ;-0.0898023954
E35D 7DB7EA 796 .BY $7D,$B7,$EA,$51,$7A
797 ; 0.110932413
E362 7D6330 798 .BY $7D,$63,$30,$88,$7E
799 ;-0.142839808
E367 7E9244 800 .BY $7E,$92,$44,$99,$3A
801 ; 0.19999912
E36C 7E4CCC 802 .BY $7E,$4C,$CC,$91,$C7
803 ;-0.333333316
E371 7FAAAA 804 .BY $7F,$AA,$AA,$AA,$13
805 ; 1
E376 810000 806 .BY $81,$00,$00,$00,$00

```

```

808 ;warm start entry (stop/restore)
809 ;
E37B 20CCFF 810 JSR SFFCC ;close all files, set default devices
E37E A900 811 LDA $00
E380 8513 812 STA Z13 ;clear CMD file number flag
E382 207AA6 813 JSR XA67A ;reset pointers
E385 58 814 CLI ;allow IRQ
E386 A280 815 BE386 LDX $80 ;index default message READY
E388 6C0003 816 JMP (X0300) ;print message (normally E38B)
817 ;
818 ;handle error messages
819 ;
E38B 8A 820 WE38B TXA
E38C 3003 821 BMI BE391 ;if X < $80
E38E 4C3AA4 822 JMP XA43A ;print BASIC message
823 ;
E391 4C74A4 824 BE391 JMP XA474 ;else print READY message
825 ;
826 ;RESET routine
827 ;
E394 2053E4 828 JSR SE453 ;initialize O/S vectors
E397 20BFE3 829 JSR SE3BF ;initialize BASIC interpreter
E39A 2022E4 830 JSR SE422 ;print BASIC start-up messages
E39D A2FB 831 LDX $FB
E39F 9A 832 TXS ;initialize stack pointer
E3A0 DOE4 833 BNE BE386 ;print message READY
834 ;
835 ;character fetch code for page zero 0073-008F
836 ;
E3A2 E67A 837 TE3A2 INC Z7A ;bump character pointer (low)
E3A4 D002 838 BNE BE3A8 ;if overflow,
E3A6 E67B 839 INC Z7B ;bump character pointer (high)
E3A8 AD60EA 840 BE3A8 LDA BEA61-1 ;fetch character from modified address
E3AB C93A 841 CMP #9+1 ;if above numerics
E3AD B00A 842 BCS BE3B9 ;return with C = 1 (and Z = 1 if ":")
E3AF C920 843 CMP $20 ;if character is a space,
E3B1 F0EF 844 BEQ TE3A2 ;ignore and get next character
E3B3 38 845 SEC
E3B4 E930 846 SBC #0 ;if below numerics
E3B6 38 847 SEC ;return with C = 1 (and Z = 1 if $00)
E3B7 E9D0 848 SBC $D0 ;if numeric, return with C = 0
E3B9 60 849 BE3B9 RTS ;return with flags set and char in A
850 ;
851 ;first RND seed value
852 ;
E3BA 804FC7 853 .BY $80,$4F,$C7,$52,$58

```

```

855 ;initialization for BASIC interpreter
856 ;
E3BF A94C 857 SE3BF LDA $4C ;set JMP instruction
E3C1 8554 858 STA Z54 ;for functions
E3C3 8D1003 859 STA X0310 ;and USR vector
E3C6 A948 860 LDA <WB248 ;default USR jump address
E3C8 A0B2 861 LDY >WB248 ;(ILLEGAL QUANTITY Error)
E3CA 8D1103 862 STA X0311 ;set USR jump address, low byte
E3CD 8C1203 863 STY X0312 ;USR jump address, high byte
E3D0 A991 864 LDA <WB391
E3D2 A0B3 865 LDY >WB391
E3D4 8505 866 STA Z05 ;set fixed-float vector
E3D6 8406 867 STY Z06 ;& high byte
E3D8 A9AA 868 LDA <WB1AA
E3DA A0B1 869 LDY >WB1AA
E3DC 8503 870 STA Z03 ;set float-fixed vector
E3DE 8404 871 STY Z04 ;& high byte
E3E0 A21C 872 LDX $1C
E3E2 BDA2E3 873 BE3E2 LDA TE3A2,X ;move character fetch code to $73-8A
E3E5 9573 874 STA Z73,X ;and first RND seed into Z8B-Z8F
E3E7 CA 875 DEX
E3E8 10F8 876 BPL BE3E2
E3EA A903 877 LDA $03
E3EC 8553 878 STA Z53 ;set default step
E3EE A900 879 LDA $00
E3F0 8568 880 STA Z68 ;clear flp accu overflow area
E3F2 8513 881 STA Z13 ;set CMD file number to default
E3F4 8518 882 STA Z18 ;clear high of string descriptor index
E3F6 A201 883 LDX $01
E3F8 8EFD01 884 STX X01FD
E3FB 8EFC01 885 STX X01FC ;prefix input buffer with dummy pointer
E3FE A219 886 LDX $19
E400 8616 887 STX Z16 ;set pointer to temporary string stack
E402 38 888 SEC
E403 209CFF 889 JSR SFF9C ;read bottom of memory address
E406 862B 890 STX Z2B ;to set BASIC bottom of memory
E408 842C 891 STY Z2C
E40A 38 892 SEC
E40B 2099FF 893 JSR SFF99 ;read top of memory address
E40E 8637 894 STX Z37 ;to set BASIC memory limit
E410 8438 895 STY Z38
E412 8633 896 STX Z33 ;set string storage address
E414 8434 897 STY Z34 ;& high byte
E416 A000 898 LDY $00
E418 98 899 TYA
E419 912B 900 STA (Z2B),Y ;move a 0 to bottom of memory
E41B E62B 901 INC Z2B ;add 1 to start of BASIC address
E41D D002 902 BNE BE421
E41F E62C 903 INC Z2C ;& high byte upon overflow
E421 60 904 BE421 RTS

```

```

906 ;print BASIC start-up messages
907 ;
E422 A52B 908 SE422 LDA Z2B ;set AY to start of BASIC
E424 A42C 909 LDY Z2C
E426 2008A4 910 JSR XA408 ;perform array area overflow check
E429 A973 911 LDA <TE473 ;set AY to message COMMODORE 64 BASIC...
E42B A0E4 912 LDY >TE473
E42D 201EAB 913 JSR XAB1E ;print message
E430 A537 914 LDA Z37 ;use limit of memory
E432 38 915 SEC
E433 E52B 916 SBC Z2B ;- pointer to start of BASIC
E435 AA 917 TAX
E436 A538 918 LDA Z38 ;& high bytes
E438 E52C 919 SBC Z2C ;to calculate bytes free
E43A 20CDBD 920 JSR XBDCC ;convert to string
E43D A960 921 LDA <TE460 ;set AY to message BYTES FREE
E43F A0E4 922 LDY >TE460
E441 201EAB 923 JSR XAB1E ;print # then BYTES FREE
E444 4C44A6 924 JMP XA644 ;perform NEW, print READY
925 ;
926 ;vectors for $0300-$030B
927 ;
E447 8BE3 928 TE447 .W WE38B ;error message vector
E449 83A4 929 .W WA483 ;warm start vector
E44B 7CA5 930 .W WA57C ;crunch BASIC tokens vector
E44D 1AA7 931 .W WA71A ;print tokens vector
E44F E4A7 932 .W WA7E4 ;execute a statement vector
E451 86AE 933 .W WAE86 ;get arithmetic element vector
934 ;
935 ;initialize vectors at $0300-$030B
936 ;
E453 A20B 937 SE453 LDX $0B
E455 BD47E4 938 BE455 LDA TE447,X ;fetch a byte from table
E458 9D0003 939 STA X0300,X ;and move it to vector area
E45B CA 940 DEX
E45C 10F7 941 BPL BE455 ;repeat until 6 vectors stored
E45E 60 942 RTS
943 ;
E45F 00 944 .BY $00
E460 204241 945 TE460 .BY " ,B,A,S,I,C, ,B,Y,T,E,S, ,F,R,E,E
E471 0D00 946 .BY $0D,$00
E473 930D20 947 TE473 .BY $93,$0D," , , , ,
E479 2A2A2A 948 .BY " , , , , ,C,O,M,M,O,D,O,R,E
E487 203634 949 .BY " ,6,4, ,B,A,S,I,C, ,V,2
E493 202A2A 950 .BY " , , , , , $0D,$0D
E49A 203634 951 .BY " ,6,4,K, ,R,A,M, ,S,Y,S,T,E,M, ,
E4AB 005C 952 .BY $00,$5C

```

```

          954 ;set output device (patch)
          955 ;
E4AD 48    956 SE4AD PHA
E4AE 20C9FF 957     JSR SFFC9      ;perform actual open for output
E4B1 AA    958     TAX            ;save possible error code
E4B2 68    959     PLA            ;restore entry accumulator
E4B3 9001   960     BCC BE4B6      ;if an error encountered,
E4B5 8A    961     TXA            ;return with error number in A
E4B6 60    962 BE4B6 RTS
          963 ;
          964 ;unused area follows
          965 ;
E4B7 AAAAAA 966     .BY $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
E4BF AAAAAA 967     .BY $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
E4C7 AAAAAA 968     .BY $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
E4CF AAAAAA 969     .BY $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
E4D7 AAAAAA 970     .BY $AA,$AA,$AA
          971 ;
          972 ;clear a byte in color RAM
          973 ;
E4DA AD21D0 974 SE4DA LDA XD021      ;use background # 0 color
E4DD 91F3   975     STA (ZF3),Y    ;to clear color RAM
E4DF 60    976     RTS
          977 ;
          978 ;pause after finding a file on cassette
          979 ;
E4E0 6902   980 SE4E0 ADC $02        ;set maximum pause to 256 - 512 jiffies
E4E2 A491   981 BE4E2 LDY Z91        ;check keyboard scan result
E4E4 C8    982     INY
E4E5 D004   983     BNE BE4EB        ;exit if a key depressed
E4E7 C5A1   984     CMP ZA1
E4E9 D0F7   985     BNE BE4E2        ;repeat until delay complete
E4EB 60    986 BE4EB RTS

```



```

988 ;baud rate factor table for International machines
989 ;((clock frequency/baud rate/2)-100)
990 ;
E4EC 1926 991 TE4EC .WO $2619 ;50 baud
E4EE 4419 992 .WO $1944 ;75
E4F0 1A11 993 .WO $111A ;110
E4F2 E80D 994 .WO $0DE8 ;134.5
E4F4 700C 995 .WO $0C70 ;150
E4F6 0606 996 .WO $0606 ;300
E4F8 D102 997 .WO $02D1 ;600
E4FA 3701 998 .WO $0137 ;1200
E4FC AE00 999 .WO $00AE ;1800
E4FE 6900 1000 .WO $0069 ;2400
1001 ;
1002 ;read base address of I/O devices into XY
1003 ;
E500 A200 1004 JE500 LDX $00
E502 A0DC 1005 LDY $DC ;point to CIA1 in XY
E504 60 1006 RTS
1007 ;
1008 ;read screen organization into XY
1009 ;
E505 A228 1010 JE505 LDX $28 ;# columns
E507 A019 1011 LDY $19 ;# rows
E509 60 1012 RTS
1013 ;
1014 ;read/set XY cursor position
1015 ;
E50A B007 1016 JE50A BCS BE513 ;if carry set, read cursor position
E50C 86D6 1017 STX ZD6 ;set cursor line # from X
E50E 84D3 1018 STY ZD3 ;set position of cursor on line from Y
E510 206CE5 1019 JSR SE56C ;set address of current screen line
E513 A6D6 1020 BE513 LDX ZD6 ;read cursor line number into X
E515 A4D3 1021 LDY ZD3 ;read position of cursor on line into Y
E517 60 1022 RTS

```

```

1024 ;initialize screen and keyboard
1025 ;
E518 20A0E5 1026 SE518 JSR SE5A0 ;initialize video chip, set default I/O
E51B A900 1027 LDA $00
E51D 8D9102 1028 STA X0291 ;enable shift mode
E520 85CF 1029 STA ZCF ;clear cursor blink phase
E522 A948 1030 LDA <WEB48
E524 8D8F02 1031 STA X028F ;set address of keyboard decode routine
E527 A9EB 1032 LDA >WEB48
E529 8D9002 1033 STA X0290 ;& high byte
E52C A90A 1034 LDA $0A
E52E 8D8902 1035 STA X0289 ;set maximum length of keyboard buffer
E531 8D8C02 1036 STA X028C ;set repeat key delay counter
E534 A90E 1037 LDA $0E
E536 8D8602 1038 STA X0286 ;set current color code
E539 A904 1039 LDA $04
E53B 8D8B02 1040 STA X028B ;set repeat key frequency counter
E53E A90C 1041 LDA $0C
E540 85CD 1042 STA ZCD ;set cursor flash timer
E542 85CC 1043 STA ZCC ;disable cursor flash
E544 AD8802 1044 SE544 LDA X0288 ;fetch screen memory start page
E547 0980 1045 ORA $80 ;set bit 7 to indicate 40 character line
E549 A8 1046 TAY
E54A A900 1047 LDA $00
E54C AA 1048 TAX
E54D 94D9 1049 BE54D STY ZD9,X ;build screen line address table
E54F 18 1050 CLC
E550 6928 1051 ADC $28
E552 9001 1052 BCC BE555
E554 C8 1053 INY
E555 E8 1054 BE555 INX
E556 E01A 1055 CPX $1A
E558 D0F3 1056 BNE BE54D ;repeat for 25 lines
E55A A9FF 1057 LDA $FF
E55C 95D9 1058 STA ZD9,X
E55E A218 1059 LDY $18 ;line count - 1
E560 20FFE9 1060 BE560 JSR SE9FF ;clear a line
E563 CA 1061 DEX
E564 10FA 1062 BPL BE560 ;loop till screen cleared
E566 A000 1063 JE566 LDY $00
E568 84D3 1064 STY ZD3 ;cursor at position 0
E56A 84D6 1065 STY ZD6 ;and line 0

```

	1067	;set address of current screen line	
	1068	;	
E56C A6D6	1069	SE56C LDX ZD6	;get cursor line number in X
E56E A5D3	1070	LDA ZD3	;and position of cursor on line in A
E570 B4D9	1071	BE570 LDY ZD9,X	;check screen line address table
E572 3008	1072	BMI BE57C	
E574 18	1073	CLC	;if on an 80 character line,
E575 6928	1074	ADC \$28	;add 40 to
E577 85D3	1075	STA ZD3	;position of cursor on line
E579 CA	1076	DEX	;point to preceeding line
E57A 10F4	1077	BPL BE570	;and repeat if screen top not reached
E57C B5D9	1078	BE57C LDA ZD9,X	;fetch high byte of screen line address
E57E 2903	1079	AND \$03	;strip 40/80 character line flag
E580 0D8802	1080	ORA X0288	;add screen memory page
E583 85D2	1081	STA ZD2	;to set high of pointer to screen line
E585 BDF0EC	1082	LDA TECF0,X	;use line number and table of low bytes
E588 85D1	1083	STA ZD1	;to set low byte of ptr to screen line
E58A A927	1084	LDA \$27	;set initial max length of current line
E58C E8	1085	INX	
E58D B4D9	1086	BE58D LDY ZD9,X	
E58F 3006	1087	BMI BE597	;exit if on a 40 character line
E591 18	1088	CLC	
E592 6928	1089	ADC \$28	;else add 40
E594 E8	1090	INX	
E595 10F6	1091	BPL BE58D	;and repeat one time
E597 85D5	1092	BE597 STA ZD5	;save maximum length current line
E599 60	1093	RTS	
	1094	;	
E59A 20A0E5	1095	JSR SE5A0	;not referenced!!
E59D 4C66E5	1096	JMP JE566	

```

1098 ;set video chip to std values and I/O devices to default
1099 ;
E5A0 A903 1100 SE5A0 LDA $03
E5A2 859A 1101 STA Z9A ;set output device (screen)
E5A4 A900 1102 LDA $00
E5A6 8599 1103 STA Z99 ;set input device (keyboard)
E5A8 A22F 1104 LDX $2F
E5AA BDB8EC 1105 BE5AA LDA TECB9-1,X ;use standard table values to
E5AD 9DFFCF 1106 STA XD000-1,X ;initialize video chip
E5B0 CA 1107 DEX
E5B1 D0F7 1108 BNE BE5AA ;repeat to move all 47 values
E5B3 60 1109 RTS
1110 ;
1111 ;fetch a character from keyboard buffer
1112 ;
E5B4 AC7702 1113 SE5B4 LDY X0277 ;save bottom char of keyboard buffer
E5B7 A200 1114 LDX $00
E5B9 BD7802 1115 BE5B9 LDA X0277+1,X ;move keyboard queue down one
E5BC 9D7702 1116 STA X0277,X
E5BF E8 1117 INX
E5C0 E4C6 1118 CPX ZC6 ;check length of queue
E5C2 D0F5 1119 BNE BE5B9 ;loop until all moved
E5C4 C6C6 1120 DEC ZC6 ;subtract one from length of queue
E5C6 98 1121 TYA ;return with character in A
E5C7 58 1122 CLI ;allow IRQ's again
E5C8 18 1123 CLC ;clear error flag
E5C9 60 1124 RTS

```

```

1126 ;wait for a Return from keyboard
1127 ;
E5CA 2016E7 1128 BE5CA JSR SE716 ;echo character on screen
E5CD A5C6 1129 BE5CD LDA ZC6 ;move keyboard count
E5CF 85CC 1130 STA ZCC ;to cursor enable flag (00=flash)
E5D1 8D9202 1131 STA X0292 ;and auto scroll down flag
E5D4 F0F7 1132 BEQ BE5CD ;loop until something arrives from kbd
E5D6 78 1133 SEI
E5D7 A5CF 1134 LDA ZCF ;if cursor blink phase 1
E5D9 F00C 1135 BEQ BE5E7
E5DB A5CE 1136 LDA ZCE ;get character under cursor
E5DD AE8702 1137 LDX X0287 ;& color of character under cursor
E5E0 A000 1138 LDY $00
E5E2 84CF 1139 STY ZCF ;clear cursor blink phase
E5E4 2013EA 1140 JSR SEA13 ;store character on screen
E5E7 20B4E5 1141 BE5E7 JSR SE5B4 ;fetch a character from the keyboard buffer
E5EA C983 1142 CMP $83 ;check for shift of Run/Stop key
E5EC D010 1143 BNE BE5FE
E5EE A209 1144 LDX $09 ;if so, set count for move
E5F0 78 1145 SEI
E5F1 86C6 1146 STX ZC6 ;force keyboard buffer count
E5F3 BDE6EC 1147 BE5F3 LDA TECE7-1,X ;move "Load/Run" into keyboard buffer
E5F6 9D7602 1148 STA X0277-1,X
E5F9 CA 1149 DEX
E5FA D0F7 1150 BNE BE5F3 ;repeat until all characters moved
E5FC F0CF 1151 BEQ BE5CD ;go get first char from forced string
1152 ;
E5FE C90D 1153 BE5FE CMP $0D ;if Return,
E600 D0C8 1154 BNE BE5CA
E602 A4D5 1155 LDY ZD5 ;move screen line length
E604 84D0 1156 STY ZD0 ;to save area
E606 B1D1 1157 BE606 LDA (ZD1),Y ;check for blanks at end of line
E608 C920 1158 CMP $20
E60A D003 1159 BNE BE60F
E60C 88 1160 DEY ;and remove them
E60D D0F7 1161 BNE BE606
E60F C8 1162 BE60F INY
E610 84C8 1163 STY ZC8 ;set end of line pointer for input
E612 A000 1164 LDY $00
E614 8C9202 1165 STY X0292 ;enable auto scroll flag
E617 84D3 1166 STY ZD3 ;set position of cursor on line
E619 84D4 1167 STY ZD4 ;clear string flag
E61B A5C9 1168 LDA ZC9 ;get saved screen line number
E61D 301B 1169 BMI BE63A ;if valid
E61F A6D6 1170 LDX ZD6
E621 20EDE6 1171 JSR SE6ED ;and equal to original
E624 E4C9 1172 CPX ZC9
E626 D012 1173 BNE BE63A
E628 A5CA 1174 LDA ZCA
E62A 85D3 1175 STA ZD3 ;set char position to original value
E62C C5C8 1176 CMP ZC8 ;if char position is past end of line,
E62E 900A 1177 BCC BE63A ;get character from screen
E630 B02B 1178 BCS BE65D ;else exit with a return

```

```

1180 ;get character from device 0 or 3 (keyboard or screen)
1181 ;
E632 98 1182 JE632 TYA
E633 48 1183 PHA
E634 8A 1184 TXA ;save XY on stack
E635 48 1185 PHA
E636 A5D0 1186 LDA ZD0 ;test saved screen line length
E638 F093 1187 BEQ BE5CD ;if zero, wait for return
1188 ;
1189 ;get character from current screen line
1190 ;
E63A A4D3 1191 BE63A LDY ZD3 ;get position of cursor on current line
E63C B1D1 1192 LDA (ZD1),Y ;then current screen character
E63E 85D7 1193 STA ZD7 ;and save it
E640 293F 1194 AND $3F ;save low order 6 bits
E642 06D7 1195 ASL ZD7 ;if bit 6 is set in original
E644 24D7 1196 BIT ZD7
E646 1002 1197 BPL BE64A
E648 0980 1198 ORA $80 ;set bit 7 in converted character
E64A 9004 1199 BE64A BCC BE650 ;if bit 7 is set in original
E64C A6D4 1200 LDX ZD4 ;and in a string
E64E D004 1201 BNE BE654 ;then don't change bit 6
E650 7002 1202 BE650 BVS BE654 ;if bit 5 is clear in original
E652 0940 1203 ORA $40 ;set bit 6 in converted character
E654 E6D3 1204 BE654 INC ZD3 ;increment position of cursor on line
E656 2084E6 1205 JSR SE684 ;check for a quote
E659 C4C8 1206 CPY ZC8 ;if end of line reached
E65B D017 1207 BNE BE674
E65D A900 1208 BE65D LDA $00
E65F 85D0 1209 STA ZD0 ;clear copy of screen line length
E661 A90D 1210 LDA $0D ;force a Return
E663 A699 1211 LDX Z99
E665 E003 1212 CPX $03 ;if input device is 3 (screen)
E667 F006 1213 BEQ BE66F ;echo character on screen
E669 A69A 1214 LDX Z9A
E66E E003 1215 CPX $03 ;if output device is screen
E66D F003 1216 BEQ BE672 ;replace character with a Return
E66F 2016E7 1217 BE66F JSR SE716 ;echo character on screen
E672 A90D 1218 BE672 LDA $0D ;replace character with a Return
E674 85D7 1219 BE674 STA ZD7 ;and save as last character
E676 68 1220 PLA
E677 AA 1221 TAX
E678 68 1222 PLA
E679 A8 1223 TAY ;restore XY
E67A A5D7 1224 LDA ZD7
E67C C9DE 1225 CMP $DE ;if character is PI
E67E D002 1226 BNE BE682
E680 A9FF 1227 LDA $FF ;return with code for PI
E682 18 1228 BE682 CLC
E683 60 1229 RTS

```

```

1231 ;check for a quote mark and set flag
1232 ;
E684 C922 1233 SE684 CMP "" ;if character is a quote
E686 D008 1234 BNE BE690
E688 A5D4 1235 LDA ZD4
E68A 4901 1236 EOR $01 ;complement string flag
E68C 85D4 1237 STA ZD4
E68E A922 1238 LDA "" ;and restore quote code in A
E690 60 1239 BE690 RTS
1240 ;
1241 ;fill screen at current position
1242 ;
E691 0940 1243 JE691 ORA $40 ;map shifted characters
E693 A6C7 1244 JE693 LDX ZC7 ;if reverse switch is on
E695 F002 1245 BEQ BE699
E697 0980 1246 JE697 ORA $80 ;add reverse video bit
E699 A6D8 1247 BE699 LDX ZD8 ;if any pending inserts
E69B F002 1248 BEQ BE69F
E69D C6D8 1249 DEC ZD8 ;decrement pending inserts count
E69F AE8602 1250 BE69F LDX X0286 ;fetch current color code
E6A2 2013EA 1251 JSR SE6A15 ;add character to screen
E6A5 20B6E6 1252 JSR SE6B6 ;get/insert new line
1253 ;
1254 ;return from output to the screen
1255 ;
E6A8 68 1256 BE6A8 PLA
E6A9 A8 1257 TAY
E6AA A5D8 1258 LDA ZD8 ;if any pending inserts,
E6AC F002 1259 BEQ BE6BE0
E6AE 46D4 1260 LSR ZD4 ;reset string mode flag
E6B0 68 1261 BE6B0 PLA
E6B1 AA 1262 TAX ;restore XA
E6B2 68 1263 PLA
E6B3 18 1264 CLC
E6B4 58 1265 CLI ;allow IRQ's
E6B5 60 1266 RTS

```

```

1268 ;get/insert new line
1269 ;
E6B6 20B3E8 1270 SE6B6 JSR SE8B3 ;check for end of a screen line
E6B9 E6D3 1271 INC ZD3 ;advance cursor position
E6BB A5D5 1272 LDA ZD5
E6BD C5D3 1273 CMP ZD3 ;if beyond maximum position,
E6BF B03F 1274 BCS BE700 ;exit
E6C1 C94F 1275 CMP $4F ;if not at 79
E6C3 F032 1276 BEQ BE6F7
E6C5 AD9202 1277 LDA X0292 ;and in auto scroll mode
E6C8 F003 1278 BEQ BE6CD ;continue
E6CA 4C67E9 1279 JMP JE967 ;else insert a blank line in screen RAM
1280 ;
E6CD A6D6 1281 BE6CD LDX ZD6 ;if in auto scroll mode
E6CF E019 1282 CPX $19 ;and cursor is on line 25
E6D1 9007 1283 BCC BE6DA
E6D3 20EAE8 1284 JSR SE8EA ;scroll screen
E6D6 C6D6 1285 DEC ZD6 ;subtract 1 from cursor line #
E6D8 A6D6 1286 LDX ZD6
E6DA 16D9 1287 BE6DA ASL ZD9,X ;set an 80 character line
E6DC 56D9 1288 LSR ZD9,X
E6DE E8 1289 INX
E6DF B5D9 1290 LDA ZD9,X ;set following line to 40 characters
E6E1 0980 1291 ORA $80
E6E3 95D9 1292 STA ZD9,X
E6E5 CA 1293 DEX
E6E6 A5D5 1294 LDA ZD5
E6E8 18 1295 CLC
E6E9 6928 1296 ADC $28 ;add 40 to line length of current line
E6EB 85D5 1297 STA ZD5
E6ED B5D9 1298 SE6ED LDA ZD9,X
E6EF 3003 1299 BMI BE6F4 ;if current line is not 80 char line
E6F1 CA 1300 DEX ;decrement line count
E6F2 D0F9 1301 BNE SE6ED ;and repeat
E6F4 4CFOE9 1302 BE6F4 JMP SE9F0 ;reset screen line address and return
1303 ;
E6F7 C6D6 1304 BE6F7 DEC ZD6 ;decrement cursor line #
E6F9 207CE8 1305 JSR SE87C ;set next line number
E6FC A900 1306 LDA $00
E6FE 85D3 1307 STA ZD3 ;set cursor position on line to 0
E700 60 1308 BE700 RTS
1309 ;
1310 ;move backwards over a line boundary
1311 ;
E701 A6D6 1312 SE701 LDX ZD6 ;if at top of screen
E703 D006 1313 BNE BE70B
E705 86D3 1314 STX ZD3 ;set cursor position to line 0
E707 68 1315 PLA
E708 68 1316 PLA ;remove own return address
E709 D09D 1317 BNE BE6A8 ;and exit
E70B CA 1318 BE70B DEX ;else back up one line
E70C 86D6 1319 STX ZD6
E70E 206CE5 1320 JSR SE56C ;set address of current screen line
E711 A4D5 1321 LDY ZD5 ;move line length of current line
E713 84D3 1322 STY ZD3 ;to position of cursor on line
E715 60 1323 RTS

```


	1325	;put a character to screen (device 3)	
	1326	;	
E716 48	1327	SE716 PHA	;save character on stack
E717 85D7	1328	STA ZD7	;and in temporary field
E719 8A	1329	TXA	
E71A 48	1330	PHA	
E71B 98	1331	TYA	
E71C 48	1332	PHA	;save XY
E71D A900	1333	LDA \$00	
E71F 85D0	1334	STA ZD0	;set saved screen line length to 0
E721 A4D3	1335	LDY ZD3	;get character position
E723 A5D7	1336	LDA ZD7	;and character
E725 1003	1337	BPL BE72A	
E727 4CD4E7	1338	JMP JE7D4	;if shifted character, skip following
	1339	;	
E72A C90D	1340	BE72A CMP \$0D	;if Return
E72C D003	1341	BNE BE731	
E72E 4C91E8	1342	JMP JE891	;perform Return function
	1343	;	
E731 C920	1344	BE731 CMP \$20	;if printable character
E733 9010	1345	BCC BE745	
E735 C960	1346	CMP \$60	
E737 9004	1347	BCC BE73D	
E739 29DF	1348	AND \$DF	
E73B D002	1349	BNE BE73F	;and not a space
E73D 293F	1350	BE73D AND \$3F	;map to screen code
E73F 2084E6	1351	BE73F JSR SE684	;check for a quote
E742 4C93E6	1352	JMP JE693	;and fill screen
	1353	;	
E745 A6D8	1354	BE745 LDX ZD8	;if non-printable
E747 F003	1355	BEQ BE74C	;and inserts pending
E749 4C97E6	1356	JMP JE697	;do not use code
	1357	\	
E74C C914	1358	BE74C CMP \$14	;if Delete code
E74E D02E	1359	BNE BE77E	
E750 98	1360	TYA	
E751 D006	1361	BNE BE759	;and not beyond left margin
E753 2001E7	1362	JSR SE701	;go backwards over line boundary
E756 4C73E7	1363	JMP JE773	;insert a blank
	1364	;	
E759 20A1E8	1365	BE759 JSR SE8A1	;check for cursor at line beginning
E75C 88	1366	DEY	
E75D 84D3	1367	STY ZD3	;adjust cursor position on line
E75F 2024EA	1368	JSR SEA24	;set color memory address
E762 C8	1369	BE762 INY	
E763 B1D1	1370	LDA (ZD1),Y	;compress screen line
E765 88	1371	DEY	
E766 91D1	1372	STA (ZD1),Y	
E768 C8	1373	INY	
E769 B1F3	1374	LDA (ZF3),Y	;and color memory
E76B 88	1375	DEY	
E76C 91F3	1376	STA (ZF3),Y	
E76E C8	1377	INY	
E76F C4D5	1378	CPY ZD5	;up to end of line
E771 D0EF	1379	BNE BE762	
E773 A920	1380	JE773 LDA \$20	;and insert blank
E775 91D1	1381	STA (ZD1),Y	;at end
E777 AD8602	1382	LDA X0286	;insert current color code
E77A 91F3	1383	STA (ZF3),Y	;in color memory
E77C 104D	1384	BPL BE7CB	

E77E	A6D4	1385	BE77E	LDX ZD4	;if in string mode
E780	F003	1386		BEQ BE785	;don't use control functions
E782	4C97E6	1387		JMP JE697	
		1388		;	
E785	C912	1389	BE785	CMP \$12	;if Reverse key
E787	D002	1390		BNE BE78B	
E789	85C7	1391		STA ZC7	;set reverse video switch
E78B	C913	1392	BE78B	CMP \$13	;if Home key
E78D	D003	1393		BNE BE792	
E78F	2066E5	1394		JSR JE566	;re-initialize cursor position
E792	C91D	1395	BE792	CMP \$1D	;if Cursor Right
E794	D017	1396		BNE BE7AD	
E796	C8	1397		INY	
E797	20B3E8	1398		JSR SE8B3	;increment character position
E79A	84D3	1399		STY ZD3	
E79C	88	1400		DEY	
E79D	C4D5	1401		CPY ZD5	;if at line end
E79F	9009	1402		BCC BE7AA	
E7A1	C6D6	1403		DEC ZD6	;decrement cursor line number
E7A3	207CE8	1404		JSR SE87C	;go to next line
E7A6	A000	1405		LDY \$00	
E7A8	84D3	1406	BE7A8	STY ZD3	;set position of cursor on line
E7AA	4CA8E6	1407	BE7AA	JMP BE6A8	;and exit
		1408		;	
E7AD	C911	1409	BE7AD	CMP \$11	;if Cursor Down
E7AF	D01D	1410		BNE BE7CE	
E7B1	18	1411		CLC	
E7B2	98	1412		TYA	
E7B3	6928	1413		ADC \$28	;advance 40 positions
E7B5	A8	1414		TAY	
E7B6	E6D6	1415		INC ZD6	;increment cursor line number
E7B8	C5D5	1416		CMP ZD5	;if not beyond line end
E7BA	90EC	1417		BCC BE7A8	
E7BC	FOEA	1418		BEQ BE7A8	;exit
E7BE	C6D6	1419		DEC ZD6	;else back up one line
E7C0	E928	1420	BE7C0	SBC \$28	;decrease character position by 40
E7C2	9004	1421		BCC BE7C8	;exit if underflow
E7C4	85D3	1422		STA ZD3	;else reset position of cursor on line
E7C6	D0F8	1423		BNE BE7C0	
E7C8	207CE8	1424	BE7C8	JSR SE87C	;set next line number
E7CB	4CA8E6	1425	BE7CB	JMP BE6A8	;exit
		1426		;	
E7CE	20CBE8	1427	BE7CE	JSR SE8CB	;check for a color code
E7D1	4C44EC	1428		JMP JEC44	;check for special CHR\$ codes, return

```

1430 ;put shifted characters to screen
1431 ;
E7D4 297F 1432 JE7D4 AND $7F ;remove shift bit
E7D6 C97F 1433 CMP $7F ;if code for PI
E7D8 D002 1434 BNE BE7DC
E7DA A95E 1435 LDA $5E ;set screen code for PI
E7DC C920 1436 BE7DC CMP $20 ;if printable
E7DE 9003 1437 BCC BE7E3
E7E0 4C91E6 1438 JMP JE691 ;go fill screen
1439 ;
E7E3 C90D 1440 BE7E3 CMP $0D ;if shifted Return
E7E5 D003 1441 BNE BE7EA
E7E7 4C91E8 1442 JMP JE891 ;do Return function
1443 ;
E7EA A6D4 1444 BE7EA LDX ZD4 ;if in string mode
E7EC D03F 1445 BNE BE82D ;do not perform control functions
E7EE C914 1446 CMP $14 ;if Insert key
E7F0 D037 1447 BNE BE829
E7F2 A4D5 1448 LDY ZD5
E7F4 B1D1 1449 LDA (ZD1),Y
E7F6 C920 1450 CMP $20 ;and last character of line is a blank
E7F8 D004 1451 BNE BE7FE
E7FA C4D3 1452 CPY ZD3 ;and not also current char position
E7FC D007 1453 BNE BE805 ;then enough space on this line
E7FE C04F 1454 BE7FE CPY $4F ;if last character of line not blank
E800 F024 1455 BEQ BE826 ;and 80 character line, ignore
E802 2065E9 1456 JSR SE965 ;else insert a blank line in screen RAM
E805 A4D5 1457 BE805 LDY ZD5 ;get new maximum character position
E807 2024EA 1458 JSR SEA24 ;set color memory address
E80A 88 1459 BE80A DEY ;from new maximum character position
E80B B1D1 1460 LDA (ZD1),Y ;move characters
E80D C8 1461 INY
E80E 91D1 1462 STA (ZD1),Y ;towards end of line
E810 88 1463 DEY
E811 B1F3 1464 LDA (ZF3),Y ;plus color memory
E813 C8 1465 INY
E814 91F3 1466 STA (ZF3),Y
E816 88 1467 DEY
E817 C4D3 1468 CPY ZD3 ;up to current position
E819 D0EF 1469 BNE BE80A
E81B A920 1470 LDA $20
E81D 91D1 1471 STA (ZD1),Y ;insert a blank
E81F AD8602 1472 LDA X0286
E822 91F3 1473 STA (ZF3),Y ;insert current color in color memory
E824 E6D8 1474 INC ZD8 ;add 1 to pending inserts count
E826 4CA8E6 1475 BE826 JMP BE6A8 ;and exit
1476 ;
E829 A6D8 1477 BE829 LDX ZD8 ;if any pending inserts
E82B F005 1478 BEQ BE832
E82D 0940 1479 BE82D ORA $40 ;add screen shift bit
E82F 4C97E6 1480 JMP JE697 ;go fill screen
1481 ;
E832 C911 1482 BE832 CMP $11 ;if Cursor Up key
E834 D016 1483 BNE BE84C
E836 A6D6 1484 LDX ZD6
E838 F037 1485 BEQ BE832 ;and cursor not on top screen line
E83A C6D6 1486 DEC ZD6
E83C A5D3 1487 LDA ZD3
E83E 38 1488 SEC
E83F E928 1489 SBC $28

```

E841	9004	1490	BCC BE847	;and current cursor position >= 40
E843	85D3	1491	STA ZD3	;decrease count by 40
E845	102A	1492	BPL BE871	;and exit
E847	206CE5	1493	BE847 JSR SE56C	;set address of current screen line
E84A	D025	1494	BNE BE871	
E84C	C912	1495	BE84C CMP \$12	;if Reverse Off key
E84E	D004	1496	BNE BE854	
E850	A900	1497	LDA \$00	
E852	85C7	1498	STA ZC7	;clear reverse video switch
E854	C91D	1499	BE854 CMP \$1D	;if Cursor Left
E856	D012	1500	BNE BE86A	
E858	98	1501	TYA	
E859	F009	1502	BEQ BE864	;and cursor not beyond left margin,
E85B	20A1E8	1503	JSR SE8A1	;check for left edge of a screen line
E85E	88	1504	DEY	;back up cursor one position
E85F	84D3	1505	STY ZD3	;and set cursor to new position
E861	4CA8E6	1506	JMP BE6A8	;and exit
		1507	;	
E864	2001E7	1508	BE864 JSR SE701	;go backwards over line boundary
E867	4CA8E6	1509	JMP BE6A8	;and exit
		1510	;	
E86A	C913	1511	BE86A CMP \$13	;if CLR key
E86C	D006	1512	BNE BE874	
E86E	2044E5	1513	JSR SE544	;re-initialize screen
E871	4CA8E6	1514	BE871 JMP BE6A8	;and exit
		1515	;	
E874	0980	1516	BE874 ORA \$80	;restore code to original value
E876	20CBE8	1517	JSR SE8CB	;check for a color change keystroke
E879	4C4FEC	1518	JMP JEC4F	;check for special CHR\$ values and return
		1519	;	
		1520	;set next line number	
		1521	;	
E87C	46C9	1522	SE87C LSR ZC9	;set saved line # on screen invalid
E87E	A6D6	1523	LDX ZD6	;get current line number
E880	E8	1524	BE880 INX	;and increment it
E881	E019	1525	CPX \$19	;if at end of screen
E883	D003	1526	BNE BE888	
E885	20EAE8	1527	JSR SE8EA	;scroll screen
E888	B5D9	1528	BE888 LDA ZD9,X	;if 80 character line,
E88A	10F4	1529	BPL BE880	;repeat
E88C	86D6	1530	STX ZD6	;store new line number
E88E	4C6CE5	1531	JMP SE56C	;set addr of current screen line and return

```

1533 ;action for Return
1534 ;
E891 A200 1535 JE891 LDX $00
E893 86D8 1536 STX ZD8 ;clear pending inserts count
E895 86C7 1537 STX ZC7 ;reset reverse video switch
E897 86D4 1538 STX ZD4 ;reset string flag
E899 86D3 1539 STX ZD3 ;set cursor to position 0 on line
E89B 207CE8 1540 JSR SE87C ;set next line number
E89E 4CA8E6 1541 JMP BE6A8 ;and exit
1542 ;
1543 ;move cursor to previous line if at start of a screen line
1544 ;
E8A1 A202 1545 SE8A1 LDX $02
E8A3 A900 1546 LDA $00
E8A5 C5D3 1547 BE8A5 CMP ZD3 ;if cursor is not
E8A7 F007 1548 BEQ BE8B0 ;at the left edge of a screen line,
E8A9 18 1549 CLC
E8AA 6928 1550 ADC $28 ;set A for next left edge
E8AC CA 1551 DEX
E8AD D0F6 1552 BNE BE8A5 ;and repeat a maximum of 2 times
E8AF 60 1553 RTS
1554 ;
E8B0 C6D6 1555 BE8B0 DEC ZD6 ;since cursor at left edge,
E8B2 60 1556 RTS ;move cursor to previous line
1557 ;
1558 ;move cursor to next line if at end of a screen line
1559 ;
E8B3 A202 1560 SE8B3 LDX $02
E8B5 A927 1561 LDA $27
E8B7 C5D3 1562 BE8B7 CMP ZD3 ;if cursor is not at the end of a line
E8B9 F007 1563 BEQ BE8C2
E8BB 18 1564 CLC
E8BC 6928 1565 ADC $28 ;point to next end of screen line
E8BE CA 1566 DEX
E8BF D0F6 1567 BNE BE8B7 ;and repeat a maximum of 2 times
E8C1 60 1568 RTS
1569 ;
E8C2 A6D6 1570 BE8C2 LDX ZD6 ;since cursor at right edge,
E8C4 E019 1571 CPX $19
E8C6 F002 1572 BEQ BE8CA
E8C8 E6D6 1573 INC ZD6 ;add 1 to cursor line number
E8CA 60 1574 BE8CA RTS ;if cursor not on last screen line
1575 ;
1576 ;check for a color change keystroke
1577 ;
E8CB A20F 1578 SE8CB LDX $0F
E8CD DDDAE8 1579 BE8CD CMP TE8DA,X ;compare keystroke to color codes
E8D0 F004 1580 BEQ BE8D6
E8D2 CA 1581 DEX
E8D3 10F8 1582 BPL BE8CD ;repeat for all 16 codes
E8D5 60 1583 RTS
1584 ;
E8D6 8E8602 1585 BE8D6 STX X0286 ;set current color code upon a match
E8D9 60 1586 RTS
1587 ;
E8DA 1588 TE8DA = * ;color codes
E8DA 90051C 1589 .BY $90,$05,$1C,$9F,$9C,$1E,$1F,$9E
E8E2 819596 1590 .BY $81,$95,$96,$97,$98,$99,$9A,$9B

```

```

1592 ;scroll screen
1593 ;
E8EA A5AC 1594 SE8EA LDA ZAC ;save I/O pointers on stack
E8EC 48 1595 PHA
E8ED A5AD 1596 LDA ZAD
E8EF 48 1597 PHA
E8F0 A5AE 1598 LDA ZAE
E8F2 48 1599 PHA
E8F3 A5AF 1600 LDA ZAF
E8F5 48 1601 PHA
E8F6 A2FF 1602 BE8F6 LDX $FF ;initialize line pointer
E8F8 C6D6 1603 DEC ZD6 ;move cursor to previous line
E8FA C6C9 1604 DEC ZC9 ;also input cursor
E8FC CEA502 1605 DEC X02A5 ;and position of first 40 char line
E8FF E8 1606 BE8FF INX
E900 20F0E9 1607 JSR SE9F0 ;set screen address
E903 E018 1608 CPX $18
E905 B00C 1609 BCS BE913 ;if 24 lines not moved up,
E907 BDF1EC 1610 LDA TECF0+1,X ;use screen line address table
E90A 85AC 1611 STA ZAC ;to set next source address
E90C B5DA 1612 LDA ZDA,X ;high byte also
E90E 20C8E9 1613 JSR SE9C8 ;move one line up
E911 30EC 1614 BMI BE8FF ;and repeat
E913 20FFE9 1615 BE913 JSR SE9FF ;clear last screen line
E916 A200 1616 LDX $00
E918 B5D9 1617 BE918 LDA ZD9,X
E91A 297F 1618 AND $7F ;set a line to 80 character
E91C B4DA 1619 LDY ZDA,X ;if next line is 80 characters
E91E 1002 1620 BPL BE922
E920 0980 1621 ORA $80 ;set to 40 character line
E922 95D9 1622 BE922 STA ZD9,X
E924 E8 1623 INX
E925 E018 1624 CPX $18
E927 D0EF 1625 BNE BE918 ;repeat until bottom of screen reached
E929 A5F1 1626 LDA ZF1 ;set bottom line to 40 character
E92B 0980 1627 ORA $80
E92D 85F1 1628 STA ZF1
E92F A5D9 1629 LDA ZD9 ;if top line is 80 characters,
E931 10C3 1630 BPL BE8F6 ;then scroll up again
E933 E6D6 1631 INC ZD6 ;else add 1 to cursor line number
E935 EEA502 1632 INC X02A5 ;and position of first 40 char line
E938 A97F 1633 LDA $7F
E93A 8D00DC 1634 STA XDC00
E93D AD01DC 1635 LDA XDC01
E940 C9FB 1636 CMP $FB ;if CTRL key is depressed,
E942 08 1637 PHP
E943 A97F 1638 LDA $7F
E945 8D00DC 1639 STA XDC00
E948 28 1640 PLP
E949 D00B 1641 BNE BE956
E94B A000 1642 LDY $00
E94D EA 1643 BE94D NOP ;delay to slow scrolling down
E94E CA 1644 DEX
E94F D0FC 1645 BNE BE94D
E951 88 1646 DEY
E952 D0F9 1647 BNE BE94D
E954 84C6 1648 STY ZC6 ;clear keyboard buffer count
E956 A6D6 1649 BE956 .J1; A1 ;put cursor line number in X
E958 68 1650 JE958 PLA
E959 85AF 1651 STA ZAF ;restore I/O area from stack

```

E95B	68	1652	PLA	
E95C	85AE	1653	STA ZAE	
E95E	68	1654	PLA	
E95F	85AD	1655	STA ZAD	
E961	68	1656	PLA	
E962	85AC	1657	STA ZAC	
E964	60	1658	RTS	
		1659	;	
		1660	;insert a blank line in screen RAM	
		1661	;	
E965	A6D6	1662	SE965	LDX ZD6 ;start from current cursor line position
E967	E8	1663	JE967	INX
E968	B5D9	1664	LDA	ZD9,X
E96A	10FB	1665	BPL	JE967 ;skip 80 column lines
E96C	8EA502	1666	STX	X02A5 ;save position of first 40 column line
E96F	ED18	1667	CPX	\$18 ;if line 24
E971	F00E	1668	BEQ	BE981
E973	900C	1669	BCC	BE981
E975	20EAE8	1670	JSR	SE8EA ;scroll screen
E978	AEA502	1671	LDX	X02A5 ;first 40 column line beyond cursor
E97B	CA	1672	DEX	;minus 1 = new cursor line number
E97C	C6D6	1673	DEC	ZD6 ;decrement cursor line number
E97E	4CDAE6	1674	JMP	BE6DA ;and go reset screen line pointers
		1675	;	
E981	A5AC	1676	BE981	LDA ZAC ;save I/O pointers on the stack
E983	48	1677	PHA	
E984	A5AD	1678	LDA	ZAD
E986	48	1679	PHA	
E987	A5AE	1680	LDA	ZAE
E989	48	1681	PHA	
E98A	A5AF	1682	LDA	ZAF
E98C	48	1683	PHA	
E98D	A219	1684	LDX	\$19 ;initialize to line 25
E98F	CA	1685	BE98F	DEX
E990	20F0E9	1686	JSR	SE9F0 ;set true screen address in ZD1/2
E993	ECA502	1687	CPX	X02A5 ;if not past first 40 column line,
E996	900E	1688	BCC	BE9A6
E998	F00C	1689	BEQ	BE9A6
E99A	BDEFEC	1690	LDA	TECF0-1,X ;use screen line address
E99D	85AC	1691	STA	ZAC ;to set source address
E99F	B5D8	1692	LDA	ZD8,X
E9A1	20C8E9	1693	JSR	SE9C8 ;move one line up
E9A4	30E9	1694	BMI	BE98F ;and repeat
E9A6	20FFE9	1695	BE9A6	JSR SE9FF ;clear screen line
E9A9	A217	1696	LDX	\$17
E9AB	ECA502	1697	BE9AB	CPX X02A5 ;if first 40 column line = 24 or 25,
E9AE	900F	1698	BCC	BE9BF
E9B0	B5DA	1699	LDA	ZDA,X
E9B2	297F	1700	AND	\$7F ;set next line to 40 column
E9B4	B4D9	1701	LDY	ZD9,X
E9B6	1002	1702	BPL	BE9BA ;if current line is 80 column,
E9B8	0980	1703	ORA	\$80 ;make into 40 column
E9BA	95DA	1704	BE9BA	STA ZDA,X
E9BC	CA	1705	DEX	
E9BD	DOEC	1706	BNE	BE9AB ;and repeat
E9BF	AEA502	1707	BE9BF	LDX X02A5 ;get first 40 column line beyond cursor
E9C2	20DAE6	1708	JSR	BE6DA ;adjust line pointers
E9C5	4C58E9	1709	JMP	JE958 ;and exit

```

1711 ;move one screen line
1712 ;
E9C8 2903 1713 SE9C8 AND $03 ;adjust addr to clear 40/80 column flag
E9CA 0D8802 1714 ORA X0288 ;add screen memory page
E9CD 85AD 1715 STA ZAD ;and save high part of address
E9CF 20E0E9 1716 JSR SE9E0 ;set color memory address in ZAE/F
E9D2 A027 1717 LDY $27
E9D4 B1AC 1718 BE9D4 LDA (ZAC),Y ;move screen byte
E9D6 91D1 1719 STA (ZD1),Y
E9D8 B1AE 1720 LDA (ZAE),Y ;move color memory byte
E9DA 91F3 1721 STA (ZF3),Y
E9DC 88 1722 DEY
E9DD 10F5 1723 BPL BE9D4 ;repeat for one screen line
E9DF 60 1724 RTS
1725 ;
1726 ;set color and screen addresses
1727 ;
E9E0 2024EA 1728 SE9E0 JSR SEA24 ;set color memory pointer in ZF3/4
E9E3 A5AC 1729 LDA ZAC ;move low byte of screen address
E9E5 85AE 1730 STA ZAE
E9E7 A5AD 1731 LDA ZAD ;get high byte of screen address
E9E9 2903 1732 AND $03 ;strip 40/80 column flag
E9EB 09D8 1733 ORA $D8 ;add screen page for color memory
E9ED 85AF 1734 STA ZAF ;and store result
E9EF 60 1735 RTS
1736 ;
1737 ;fetch screen address, depending on X
1738 ;
E9F0 BDF0EC 1739 SE9F0 LDA TECFO,X ;set low byte of screen address
E9F3 85D1 1740 STA ZD1
E9F5 85D9 1741 LDA ZD9,X ;fetch high byte
E9F7 2903 1742 AND $03 ;strip 40/80 column flag
E9F9 0D8802 1743 ORA X0288 ;add screen memory page
E9FC 85D2 1744 STA ZD2 ;and store result
E9FE 60 1745 RTS

```



```

1747 ;clear one screen line
1748 ;
E9FF A027 1749 SE9FF LDY $27
EA01 20F0E9 1750 JSR SE9F0 ;set true screen RAM address
EA04 2024EA 1751 JSR SEA24 ;set color memory address
EA07 A920 1752 BEA07 LDA $20
EA09 91D1 1753 STA (ZD1),Y ;clear a byte in video RAM
EA0B 20DAE4 1754 JSR SE4DA ;clear a byte in color memory
EA0E EA 1755 NOP ;(length adjustment for patch)
EA0F 88 1756 DEY
EA10 10F5 1757 BPL BEA07 ;repeat for one line
EA12 60 1758 RTS
1759 ;
1760 ;set cursor flash timing and color memory address
1761 ;
EA13 A8 1762 SEA13 TAY
EA14 A902 1763 LDA $02
EA16 85CD 1764 STA ZCD ;init cursor flash timing countdown
EA18 2024EA 1765 JSR SEA24 ;set color memory address
EA1B 98 1766 TYA
1767 ;
1768 ;store a character to the screen
1769 ;
EA1C A4D3 1770 SEA1C LDY ZD3 ;fetch position on line
EA1E 91D1 1771 STA (ZD1),Y ;move char to current screen address
EA20 8A 1772 TXA ;fetch character color
EA21 91F3 1773 STA (ZF3),Y ;move to color memory
EA23 60 1774 RTS
1775 ;
1776 ;set color memory address parallel to screen
1777 ;
EA24 A5D1 1778 SEA24 LDA ZD1 ;move low screen address
EA26 85F3 1779 STA ZF3 ;to low of color memory address
EA28 A5D2 1780 LDA ZD2 ;fetch high of screen address
EA2A 2903 1781 AND $03 ;clear 40/80 column flag
EA2C 09D8 1782 ORA $D8 ;adjust to color memory area
EA2E 85F4 1783 STA ZF4 ;set high of color memory address
EA30 60 1784 RTS

```

```

1786 ;standard IRQ entry
1787 ;
EA31 20EAF 1788 WEA31 JSR SFEEA ;bump real time clock
EA34 A5CC 1789 LDA ZCC ;check cursor flash enable flag
EA36 D029 1790 BNE BEA61 ;don't flash if non-zero
EA38 C6CD 1791 DEC ZCD ;decrement cursor flash countdown
EA3A D025 1792 BNE BEA61 ;and skip if not finished (flash delay)
EA3C A914 1793 LDA $14 ;reset jiffy count for next cursor flip
EA3E 85CD 1794 STA ZCD
EA40 A4D3 1795 LDY ZD3 ;fetch cursor position
EA42 46CF 1796 LSR ZCF ;shift cursor flash phase
EA44 AE8702 1797 LDX X0287 ;fetch color under cursor
EA47 B1D1 1798 LDA (ZD1),Y ;get character under cursor
EA49 B011 1799 BCS BEA5C ;branch if not first entry for character
EA4B E6CF 1800 INC ZCF ;set cursor flash phase to 1
EA4D 85CE 1801 STA ZCE ;save character under cursor
EA4F 2024EA 1802 JSR SEA24 ;set color memory address
EA52 B1F3 1803 LDA (ZF3),Y
EA54 8D8702 1804 STA X0287 ;save color under cursor
EA57 AE8602 1805 LDX X0286 ;get current color code
EA5A A5CE 1806 LDA ZCE ;get current character
EA5C 4980 1807 BEA5C EOR $80 ;invert character
EA5E 201CEA 1808 JSR SEA1C ;display on screen
EA61 A501 1809 BEA61 LDA Z01 ;check cassette sense line
EA63 2910 1810 AND $10
EA65 F00A 1811 BEQ BEA71 ;if no buttons pressed
EA67 A000 1812 LDY $00
EA69 84C0 1813 STY ZC0 ;clear tape motor flag
EA6B A501 1814 LDA Z01
EA6D 0920 1815 ORA $20 ;turn motor off
EA6F D008 1816 BNE BEA79 ;JMP
EA71 A5C0 1817 BEA71 LDA ZC0 ;check tape motor flag
EA73 D006 1818 BNE BEA7B
EA75 A501 1819 LDA Z01 ;if flag shows motor should be on,
EA77 291F 1820 AND $1F ;turn cassette motor on
EA79 8501 1821 BEA79 STA Z01
EA7B 2087EA 1822 BEA7B JSR SEA87 ;scan keyboard
EA7E AD0DDC 1823 LDA XDC0D ;clear any pending IRQ's in ICR1
EA81 68 1824 PLA ;restore all registers
EA82 A8 1825 TAY
EA83 68 1826 PLA
EA84 AA 1827 TAX
EA85 68 1828 PLA
EA86 40 1829 RTI ;return from IRQ

```

```

1831 ;scan keyboard
1832 ;
EA87 A900 1833 SEAB7 LDA $00
EA89 8D8D02 1834 STA X028D ;clear shift/control flag
EA8C A040 1835 LDY $40
EA8E 84CB 1836 STY ZCB ;set key pressed flag to default
EA90 8D00DC 1837 STA XDC00 ;clear row
EA93 AE01DC 1838 LDX XDC01 ;check column
EA96 E0FF 1839 CPX $FF
EA98 F061 1840 BEQ BEAFB ;if a key pressed
EA9A A8 1841 TAY ;save in Y
EA9B A981 1842 LDA <TEB81 ;set address of standard keyboard table
EA9D 85F5 1843 STA ZF5 ;in keyboard table pointer
EA9F A9EB 1844 LDA >TEB81
EAA1 85F6 1845 STA ZF6 ;& high byte
EAA3 A9FE 1846 LDA $FE
EAA5 8D00DC 1847 STA XDC00 ;set first row
EAA8 A208 1848 BEAAB LDX $08 ;column count
EAAA 48 1849 PHA ;save current row on stack
EABAB AD01DC 1850 BEAAB LDA XDC01 ;read a column
EAAE CD01DC 1851 CMP XDC01
EAB1 D0F8 1852 BNE BEAAB ;wait until steady
EAB3 4A 1853 BEAB3 LSR A
EAB4 B016 1854 BCS BEACC ;if a key is pressed,
EAB6 48 1855 PHA ;save on stack
EAB7 B1F5 1856 LDA (ZF5),Y ;fetch keyboard table contents
EAB9 C905 1857 CMP $05
EABB B00C 1858 BCS BEAC9 ;if < 5
EABD C903 1859 CMP $03 ;check for Commodore key
EABF F008 1860 BEQ BEAC9
EAC1 0D8D02 1861 ORA X028D ;add possible shift/control flag
EAC4 8D8D02 1862 STA X028D ;to set keyboard shift/control flag
EAC7 1002 1863 BPL BEACB
EAC9 84CB 1864 BEAC9 STY ZCB ;set key pressed
EACB 68 1865 BEACB PLA ;restore last code read
EACC C8 1866 BEACC INY
EACD C041 1867 CPY $41
EACF B00B 1868 BCS BEADC ;if scan not complete,
EAD1 CA 1869 DEX
EAD2 D0DF 1870 BNE BEAB3 ;check next bit in column data
EAD4 38 1871 SEC
EAD5 68 1872 PLA ;else restore row position
EAD6 2A 1873 ROL A ;select next row
EAD7 8D00DC 1874 STA XDC00
EADA D0CC 1875 BNE BEAAB ;and repeat
EADC 68 1876 BEADC PLA ;restore code
EADD 6C8F02 1877 JMP (X028F) ;perform kbd decode routine (EB48)
1878 ;
EAE0 A4CB 1879 JEAE0 LDY ZCB ;fetch key pressed
EAE2 B1F5 1880 LDA (ZF5),Y ;decode via keyboard table
EAE4 AA 1881 TAX
EAE5 C4C5 1882 CPY ZC5 ;if not the same as the last key,
EAE7 F007 1883 BEQ BEAF0
EAE9 A010 1884 LDY $10
EAEB 8C8C02 1885 STY X028C ;set repeat key delay counter
EAE EEE 1886 BNE BEB26 ;and skip repeat logic
1887 ;
EAF0 297F 1888 BEAF0 AND $7F
EAF2 2C8A02 1889 BIT X028A ;if repeat flag has
EAF5 3016 1890 BMI BEB0D ;bit 7 set, repeat all keys

```

EAF7	7049	1891	BVS	BEB42	;exit when flag > 63
EAF9	C97F	1892	CMP	\$7F	
EAFB	F029	1893	BEAFB	BEQ	BEB26
EAFD	C914	1894	CMP	\$14	;delete
EAFD	F00C	1895	BEQ	BEB0D	
EB01	C920	1896	CMP	\$20	;space
EB03	F008	1897	BEQ	BEB0D	
EB05	C91D	1898	CMP	\$1D	;cursor right/left
EB07	F004	1899	BEQ	BEB0D	
EB09	C911	1900	CMP	\$11	;cursor down/up
EB0B	D035	1901	BNE	BEB42	;if none of above, don't repeat
EB0D	AC8C02	1902	BEB0D	LDY	X028C
EB10	F005	1903	BEQ	BEB17	;check repeat key delay counter
EB12	CE8C02	1904	DEC	X028C	;countdown complete
EB15	D02B	1905	BNE	BEB42	;subtract 1 from repeat key delay ctr
EB17	CE8B02	1906	BEB17	DEC	X028B
EB1A	D026	1907	BNE	BEB42	;subtract 1 from repeat key freq ctr
EB1C	A004	1908	LDY	\$04	
EB1E	8C8B02	1909	STY	X028B	;reset repeat key frequency counter
EB21	A4C6	1910	LDY	ZC6	;check keyboard buffer count
EB23	88	1911	DEY		
EB24	101C	1912	BPL	BEB42	;branch if something in keyboard queue
EB26	A4CB	1913	BEB26	LDY	ZCB
EB28	84C5	1914	STY	ZC5	;move current key
EB2A	AC8D02	1915	LDY	X028D	;to last key
EB2D	8C8E02	1916	STY	X028E	;move shift/control flag
EB30	E0FF	1917	CPX	\$FF	;to last shift pattern
EB32	F00E	1918	BEQ	BEB42	
EB34	8A	1919	TXA		
EB35	A6C6	1920	LDX	ZC6	;check keyboard queue length
EB37	EC8902	1921	CPX	X0289	;against maximum allowed
EB3A	B006	1922	BCS	BEB42	;branch if buffer full
EB3C	9D7702	1923	STA	X0277,X	;else move character to keyboard queue
EB3F	E8	1924	INX		
EB40	86C6	1925	STX	ZC6	;add 1 to length of keyboard queue
EB42	A97F	1926	BEB42	LDA	\$7F
EB44	8D00DC	1927	STA	XDC00	;set PAL bits high
EB47	60	1928	RTS		
		1929			
EB48	AD8D02	1930	WEB48	LDA	X028D
EB4B	C903	1931	CMP	\$03	;check shift/control flag
EB4D	D015	1932	BNE	BEB64	;for Commodore key and shift depressed
EB4F	CD8E02	1933	CMP	X028E	;if so, and code is different
EB52	F0EE	1934	BEQ	BEB42	;from last keyboard shift pattern
EB54	AD9102	1935	LDA	X0291	
EB57	301D	1936	BMI	BEB76	;and not in shift lock mode
EB59	AD18D0	1937	LDA	XD018	
EB5C	4902	1938	EOR	\$02	;then toggle lower case/upper case bit
EB5E	8D18D0	1939	STA	XD018	;set video chip bit for proper display
EB61	4C76EB	1940	JMP	BEB76	;exit

```

1942 ;select new keyboard table
1943 ;
EB64 0A 1944 BEB64 ASL A ;double entry index
EB65 C908 1945 CMP $08 ;if index is too high,
EB67 9002 1946 BCC BEB6E
EB69 A906 1947 LDA $06 ;select "Control Key" keyboard
EB6B AA 1948 BEB6B TAX
EB6C BD79EB 1949 LDA TEB79,X ;fetch low byte from table
EB6F 85F5 1950 STA ZF5 ;and set keyboard table pointer
EB71 BD7AEB 1951 LDA TEB79+1,X
EB74 85F6 1952 STA ZF6 ;& high byte
EB76 4CE0EA 1953 BEB76 JMP JAEAO ;exit
1954 ;
1955 ;table of keyboard table addresses
1956 ;
EB79 81EB 1957 TEB79 .W TEB81 ;Standard keyboard
EB7B C2EB 1958 .W TEB82 ;"Shift" keyboard
EB7D 03EC 1959 .W TEC03 ;"Commodore Key" keyboard
EB7F 78EC 1960 .W TEC78 ;"Control" keyboard
1961 ;
1962 ;standard keyboard
1963 ;
EB81 140D1D 1964 TEB81 .BY $14,$0D,$1D,$88,$85,$86,$87,$11
EB89 335741 1965 .BY $33,$57,$41,$34,$5A,$53,$45,$01
EB91 355244 1966 .BY $35,$52,$44,$36,$43,$46,$54,$58
EB99 375947 1967 .BY $37,$59,$47,$38,$42,$48,$55,$56
EBA1 39494A 1968 .BY $39,$49,$4A,$30,$4D,$4B,$4F,$4E
EBA9 2B504C 1969 .BY $2B,$50,$4C,$2D,$2E,$3A,$40,$2C
EBB1 5C2A3B 1970 .BY $5C,$2A,$3B,$13,$01,$3D,$5E,$2F
EBB9 315F04 1971 .BY $31,$5F,$04,$32,$20,$02,$51,$03,$FF
1972 ;
1973 ;"Shift" keyboard table
1974 ;
EB82 948D9D 1975 TEB82 .BY $94,$8D,$9D,$8C,$89,$8A,$8B,$91
EB8A 23D7C1 1976 .BY $23,$D7,$C1,$24,$DA,$D3,$C5,$01
EBD2 25D2C4 1977 .BY $25,$D2,$C4,$26,$C3,$C6,$D4,$D8
EBDA 27D9C7 1978 .BY $27,$D9,$C7,$28,$C2,$C8,$D5,$D6
EBE2 29C9CA 1979 .BY $29,$C9,$CA,$30,$CD,$CB,$CF,$CE
EBEA DBDOCC 1980 .BY $DB,$D0,$CC,$DD,$3E,$5B,$8A,$3C
EBF2 A9C05D 1981 .BY $A9,$C0,$5D,$93,$01,$3D,$DE,$3F
EBFA 215F04 1982 .BY $21,$5F,$04,$22,$A0,$02,$D1,$83,$FF
1983 ;
1984 ;"Commodore Key" keyboard table
1985 ;
EC03 948D9D 1986 TEC03 .BY $94,$8D,$9D,$8C,$89,$8A,$8B,$91
EC0B 96B3B0 1987 .BY $96,$B3,$B0,$97,$AD,$AE,$B1,$01
EC13 98B2AC 1988 .BY $98,$B2,$AC,$99,$BC,$BB,$A3,$BD
EC1B 9AB7A5 1989 .BY $9A,$B7,$A5,$9B,$BF,$B4,$B8,$BE
EC23 29A2B5 1990 .BY $29,$A2,$B5,$30,$A7,$A1,$B9,$AA
EC2B A6AFB6 1991 .BY $A6,$AF,$B6,$DC,$3E,$5B,$A4,$3C
EC33 ABDF5D 1992 .BY $AB,$DF,$5D,$93,$01,$3D,$DE,$3F
EC3B 815F04 1993 .BY $81,$5F,$04,$95,$A0,$02,$AB,$83,$FF

```

```

1995 ;check for special CHR$ values
1996 ;
EC44 C90E 1997 JEC44 CMP $0E ;if CHR$(14) to be printed,
EC46 D007 1998 BNE JEC4F
EC48 AD18D0 1999 LDA XD018
EC4B 0902 2000 ORA $02 ;set value for unshift/shift
EC4D D009 2001 BNE BEC58 ;reset address and exit
EC4F C98E 2002 JEC4F CMP $8E ;if CHR$(142) to be printed,
EC51 D00B 2003 BNE BEC5E
EC53 AD18D0 2004 LDA XD018
EC56 29FD 2005 AND $FD ;set value for shift/graphics
EC58 8D18D0 2006 BEC58 STA XD018 ;reset video chip register
EC5B 4CABE6 2007 BEC5B JMP BE6A8 ;exit
2008 ;
EC5E C908 2009 BEC5E CMP $08 ;if CHR$(8) to be printed,
EC60 D007 2010 BNE BEC69
EC62 A980 2011 LDA $80 ;set shift mode to locked
EC64 0D9102 2012 ORA X0291
EC67 3009 2013 BMI BEC72 ;and exit
EC69 C909 2014 BEC69 CMP $09 ;if CHR$(9) to be printed,
EC6B D0EE 2015 BNE BEC5B
EC6D A97F 2016 LDA $7F ;clear shift lock flag
EC6F 2D9102 2017 AND X0291
EC72 8D9102 2018 BEC72 STA X0291 ;condition shift lock flag
EC75 4CABE6 2019 JMP BE6A8 ;exit
2020 ;
2021 ;"Control Key" keyboard table
2022 ;
EC78 FFFFFF 2023 TEC78 .BY $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
EC80 1C1701 2024 .BY $1C,$17,$01,$9F,$1A,$13,$05,$FF
EC88 9C1204 2025 .BY $9C,$12,$04,$1E,$03,$06,$14,$18
EC90 1F1907 2026 .BY $1F,$19,$07,$9E,$02,$08,$15,$16
EC98 12090A 2027 .BY $12,$09,$0A,$92,$0D,$0B,$0F,$0E
ECA0 FF100C 2028 .BY $FF,$10,$0C,$FF,$FF,$1B,$00,$FF
ECA8 1CFF1D 2029 .BY $1C,$FF,$1D,$FF,$FF,$1F,$1E,$FF
ECB0 9006FF 2030 .BY $90,$06,$FF,$05,$FF,$FF,$11,$FF,$FF

```

```

2032 ;standard values for video chip
2033 ;
ECB9 00 2034 TECB9 .BY $00 ;+ 00, MOB 0 X position
ECBA 00 2035 .BY $00 ;+ 01, MOB 0 Y position
ECBB 00 2036 .BY $00 ;+ 02, MOB 1 X position
ECBC 00 2037 .BY $00 ;+ 03, MOB 1 Y position
ECBD 00 2038 .BY $00 ;+ 04, MOB 2 X position
ECBE 00 2039 .BY $00 ;+ 05, MOB 2 Y position
ECBF 00 2040 .BY $00 ;+ 06, MOB 3 X position
ECC0 00 2041 .BY $00 ;+ 07, MOB 3 Y position
ECC1 00 2042 .BY $00 ;+ 08, MOB 4 X position
ECC2 00 2043 .BY $00 ;+ 09, MOB 4 Y position
ECC3 00 2044 .BY $00 ;+ 0A, MOB 5 X position
ECC4 00 2045 .BY $00 ;+ 0B, MOB 5 Y position
ECC5 00 2046 .BY $00 ;+ 0C, MOB 6 X position
ECC6 00 2047 .BY $00 ;+ 0D, MOB 6 Y position
ECC7 00 2048 .BY $00 ;+ 0E, MOB 7 X position
ECC8 00 2049 .BY $00 ;+ 0F, MOB 7 Y position
ECC9 00 2050 .BY $00 ;+ 10, MSB of X position
ECCA 9B 2051 .BY $9B ;+ 11, RC3, DEN, RSEL, Y1 and Y0 set
ECCB 37 2052 .BY $37 ;+ 12, Raster Register
ECCC 00 2053 .BY $00 ;+ 13, Light Pen X
ECCD 00 2054 .BY $00 ;+ 14, Light Pen Y
ECCF 08 2055 .BY $00 ;+ 15, MOB Enable
ECCF 08 2056 .BY $08 ;+ 16, CSEL set
ECD0 00 2057 .BY $00 ;+ 17, MOB Y expand
ECD1 14 2058 .BY $14 ;+ 18, Memory Pointers
ECD2 0F 2059 .BY $0F ;+ 19, Interrupt Register
ECD3 00 2060 .BY $00 ;+ 1A, Enable Interrupts
ECD4 00 2061 .BY $00 ;+ 1B, MOB-Data priority
ECD5 00 2062 .BY $00 ;+ 1C, MOB Multi-Color selection
ECD6 00 2063 .BY $00 ;+ 1D, MOB X-expand
ECD7 00 2064 .BY $00 ;+ 1E, MOB-MOB collision
ECD8 00 2065 .BY $00 ;+ 1F, MOB-Data collision
ECD9 0E 2066 .BY $0E ;+ 20, Exterior color
ECDA 06 2067 .BY $06 ;+ 21, Background # 0 color
ECDB 01 2068 .BY $01 ;+ 22, Background # 1 color
ECDC 02 2069 .BY $02 ;+ 23, Background # 2 color
ECDD 03 2070 .BY $03 ;+ 24, Background # 3 color
ECDE 04 2071 .BY $04 ;+ 25, MOB Multi-Color # 0
ECDF 00 2072 .BY $00 ;+ 26, MOB Multi-Color # 1
ECE0 01 2073 .BY $01 ;+ 27, MOB 0 color
ECE1 02 2074 .BY $02 ;+ 28, MOB 1 color
ECE2 03 2075 .BY $03 ;+ 29, MOB 2 color
ECE3 04 2076 .BY $04 ;+ 2A, MOB 3 color
ECE4 05 2077 .BY $05 ;+ 2B, MOB 4 color
ECE5 06 2078 .BY $06 ;+ 2C, MOB 5 color
ECE6 07 2079 .BY $07 ;+ 2D, MOB 6 color
2080 ;+2E, MOB 7 color initialized to $4C
2081 ;
2082 ;keyboard buffer entry for shift of Run/Stop key
2083 ;
ECE7 4C4F41 2084 TECF7 .BY "L","O","A","D",$0D
ECEC 52554E 2085 .BY "R","U","N",$0D
2086 ;
2087 ;table of low bytes of screen line addresses
2088 ;
ECF0 002850 2089 TECF0 .BY $00,$28,$50,$78,$A0,$C8,$F0,$18
ECF8 406890 2090 .BY $40,$68,$90,$B8,$E0,$08,$30,$58
ED00 80A8D0 2091 .BY $80,$A8,$D0,$F8,$20,$48,$70,$98,$C0

```

		2093	;send TALK on serial bus	
		2094	;	
ED09	0940	2095	SED09 ORA \$40	;add offset for TALK
ED0B	2C	2096	.BY \$2C	;skip next instruction
		2097	;	
		2098	;send LISTEN on serial bus	
		2099	;	
ED0C	0920	2100	SED0C ORA \$20	;add offset for LISTEN
ED0E	20A4F0	2101	JSR SFOA4	;protect against RS-232 NMI's
ED11	48	2102	SED11 PHA	;save code to transmit
ED12	2494	2103	BIT Z94	;if no data is pending
ED14	100A	2104	BPL BED20	
ED16	38	2105	SEC	
ED17	66A3	2106	ROR ZA3	;set EOI flag
ED19	2040ED	2107	JSR SED40	;output byte on serial bus
ED1C	4694	2108	LSR Z94	;clear data pending flag
ED1E	46A3	2109	LSR ZA3	;clear EOI flag
ED20	68	2110	BED20 PLA	;move code to transmit
ED21	8595	2111	STA Z95	;into the output buffer
ED23	78	2112	SEI	
ED24	2097EE	2113	JSR SEE97	;set serial bus data line low
ED27	C93F	2114	CMP \$3F	
ED29	D003	2115	BNE BED2E	;if not UNLISTEN
ED2B	2085EE	2116	JSR SEE85	;set serial clock low
ED2E	AD00DD	2117	BED2E LDA XDD00	
ED31	0908	2118	ORA \$08	
ED33	8D00DD	2119	STA XDD00	;set serial ATN line high
ED36	78	2120	SED36 SEI	;disable IRQ
ED37	208EEE	2121	JSR SEE8E	;set serial clock high
ED3A	2097EE	2122	JSR SEE97	;set serial data low
ED3D	20B3EE	2123	JSR SEEB3	;delay 1 millisecond


```

2125 ;send a byte from Z95 on serial bus
2126 ;
ED40 78 2127 SED40 SEI ;disable IRQ
ED41 2097EE 2128 JSR SEE97 ;set serial data low
ED44 20A9EE 2129 JSR SEEA9 ;test data level
ED47 B064 2130 BCS BEDAD ;if high, indicate device not present
ED49 2085EE 2131 JSR SEE85 ;set serial clock low
ED4C 24A3 2132 BIT ZA3
ED4E 100A 2133 BPL BED5A ;if EOI flag is set
ED50 20A9EE 2134 BED50 JSR SEEA9
ED53 90FB 2135 BCC BED50 ;wait for data high
ED55 20A9EE 2136 BED55 JSR SEEA9
ED58 B0FB 2137 BCS BED55 ;wait for data low
ED5A 20A9EE 2138 BED5A JSR SEEA9
ED5D 90FB 2139 BCC BED5A ;wait for data high (end of EOI pulse)
ED5F 208EEE 2140 JSR SEE8E ;set output clock high
ED62 A908 2141 LDA $08
ED64 85A5 2142 STA ZA5 ;initialize bit count
ED66 AD00DD 2143 BED66 LDA XDD00
ED69 CD00DD 2144 CMP XDD00
ED6C D0F8 2145 BNE BED66 ;wait for PA2 to steady
ED6E 0A 2146 ASL A ;if data high not received,
ED6F 903F 2147 BCC BEDB0 ;indicate a timeout
ED71 6695 2148 ROR Z95 ;fetch bit to send
ED73 B005 2149 BCS BED7A ;if bit is a 0,
ED75 20A0EE 2150 JSR SEEA0 ;send a 0
ED78 D003 2151 BNE BED7D
ED7A 2097EE 2152 BED7A JSR SEE97 ;else send a 1
ED7D 2085EE 2153 JSR SEE85 ;set serial clock low
ED80 EA 2154 NOP
ED81 EA 2155 NOP
ED82 EA 2156 NOP ;hold clock/data lines for 8 usec
ED83 EA 2157 NOP
ED84 AD00DD 2158 LDA XDD00
ED87 29DF 2159 AND $DF ;set serial data line low
ED89 0910 2160 ORA $10 ;and clock high
ED8B 8D00DD 2161 STA XDD00
ED8E C6A5 2162 DEC ZA5 ;if 8 bits not sent
ED90 D0D4 2163 BNE BED66 ;repeat
ED92 A904 2164 LDA $04
ED94 8D07DC 2165 STA XDC07 ;set TBH1 to $04xx
ED97 A919 2166 LDA $19 ;set CBI to force load, one shot and TB1
ED99 8D0FDC 2167 STA XDC0F
ED9C AD0DDC 2168 LDA XDC0D ;clear pending IRQ bits in ICRI
ED9F AD0DDC 2169 BED9F LDA XDC0D
EDA2 2902 2170 AND $02
EDA4 D00A 2171 BNE BEDB0 ;if underflow on TB, indicate
EDA6 20A9EE 2172 JSR SEEA9
EDA9 B0F4 2173 BCS BED9F ;repeat test until data line is low
EDAB 58 2174 CLI ;enable IRQ
EDAC 60 2175 RTS
2176 ;
EDAD A980 2177 BEDAD LDA $80 ;set Device Not Present
EDAF 2C 2178 .BY $2C ;skip next instruction
EDB0 A903 2179 BEDB0 LDA $03 ;set Timeout bits (read and write)
EDB2 201CFE 2180 JEDB2 JSR SFE1C ;add bits in A to ST
EDB5 58 2181 CLI
EDB6 18 2182 CLC
EDB7 904A 2183 BCC BEE03 ;set ATN low and exit

```

```

2185 ;send Secondary Address (after LISTEN) on serial bus
2186 ;
EDB9 8595 2187 SEDB9 STA Z95 ;save serial deferred character
EDBB 2036ED 2188 JSR SED36 ;send on serial bus
EDBE AD00DD 2189 SEDBE LDA XDD00
EDC1 29F7 2190 AND $F7 ;set ATN low
EDC3 8D00DD 2191 STA XDD00
EDC6 60 2192 RTS
2193 ;
2194 ;send Secondary Address (after TALK) on serial bus
2195 ;
EDC7 8595 2196 SEDC7 STA Z95 ;save serial deferred character
EDC9 2036ED 2197 JSR SED36 ;send on serial bus
EDCC 78 2198 SEDCC SEI
EDCD 20A0EE 2199 JSR SEEA0 ;set serial data high
EDDO 20BEED 2200 JSR SEDBE ;set ATN high
EDD3 2085EE 2201 JSR SEE85 ;set serial clock low
EDD6 20A9EE 2202 BEDD6 JSR SEEA9 ;read serial clock
EDD9 30FB 2203 BMI BEDD6 ;loop until serial clock goes low
EDDB 58 2204 CLI ;enable IRQ
EDDC 60 2205 RTS
2206 ;
2207 ;output byte on serial bus
2208 ;
EDDD 2494 2209 JEDDD BIT Z94
EDDF 3005 2210 BMI BEDE6 ;if no data buffered,
EDE1 38 2211 SEC
EDE2 6694 2212 ROR Z94 ;set serial deterred flag
EDE4 D005 2213 BNE BEDEB
EDE6 48 2214 BEDE6 PHA ;else save character on stack
EDE7 2040ED 2215 JSR SED40 ;send deferred character on serial bus
EDEA 68 2216 PLA
EDEB 8595 2217 BEDEB STA Z95 ;save new serial deferred character
EDED 18 2218 CLC
EDEE 60 2219 RTS
2220 ;
2221 ;send TALK on serial bus
2222 ;
EDEF 78 2223 SEDEF SEI ;disable IRQ
EDF0 208EEE 2224 JSR SEE8E ;set serial clock low
EDF3 AD00DD 2225 LDA XDD00
EDF6 0908 2226 ORA $08 ;set serial ATN high
EDF8 8D00DD 2227 STA XDD00
EDFB A95F 2228 LDA $5F ;set A to code for TALK
EDFD 2C 2229 .BY $2C ;skip next instruction
2230 ;
2231 ;send UNLISTEN on serial bus
2232 ;
EDFE A93F 2233 SEDFE LDA $3F ;set A to code for UNLISTEN
EE00 2011ED 2234 JSR SED11 ;send code in A on serial bus
EE03 20BEED 2235 BEE03 JSR SEDBE ;set ATN low
EE06 8A 2236 SEE06 TXA
EE07 A20A 2237 LDX $0A
EE09 CA 2238 BEE09 DEX
EE0A D0FD 2239 BNE BEE09 ;pause 52 microseconds
EE0C AA 2240 TAX
EE0D 2085EE 2241 JSR SEE85 ;set serial clock low
EE10 4C97EE 2242 JMP SEE97 ;set serial data low and return

```

```

2244 ;input byte on serial bus
2245 ;
EE13 78 2246 JEE13 SEI ;disable IRQ
EE14 A900 2247 LDA $00
EE16 85A5 2248 STA ZA5 ;clear bit count
EE18 2085EE 2249 JSR SEE85 ;set serial clock low
EE1B 20A9EE 2250 BEE1B JSR SEEA9 ;read serial clock
EE1E 10FB 2251 BPL BEE1B ;and loop until high
EE20 A901 2252 BEE20 LDA $01
EE22 8D07DC 2253 STA XDC07 ;prime TBH1 with $01xx
EE25 A919 2254 LDA $19
EE27 8D0FDC 2255 STA XDC0F ;set CB1 to force load, one shot & TB1
EE2A 2097EE 2256 JSR SEE97 ;set serial data low
EE2D AD0DDC 2257 LDA XDC0D ;clear pending IRQ bits in ICRI
EE30 AD0DDC 2258 BEE30 LDA XDC0D
EE33 2902 2259 AND $02
EE35 D007 2260 BNE BEE3E ;if not underflow in TB
EE37 20A9EE 2261 JSR SEEA9 ;read serial clock and data
EE3A 30F4 2262 BMI BEE30 ;repeat if clock still low
EE3C 1018 2263 BPL BEE36 ;else go receive byte
EE3E A5A5 2264 BEE3E LDA ZA5 ;when underflow on TB and not first bit,
EE40 F005 2265 BEQ BEE47
EE42 A902 2266 LDA $02 ;set read timeout in ST
EE44 4CB2ED 2267 JMP JEDB2
2268 ;
EE47 20A0EE 2269 BEE47 JSR SEEA0 ;set serial data high
EE4A 2085EE 2270 JSR SEE85 ;set serial clock low
EE4D A940 2271 LDA $40
EE4F 201CFE 2272 JSR SFE1C ;set EOI in ST
EE52 E6A5 2273 INC ZA5 ;add one to bit count
EE54 D0CA 2274 BNE BEE20 ;and repeat
EE56 A908 2275 BEE56 LDA $08
EE58 85A5 2276 STA ZA5 ;set bit count to 8
EE5A AD00DD 2277 BEE5A LDA XDD00
EE5D CD00DD 2278 CMP XDD00
EE60 D0F8 2279 BNE BEE5A ;wait until PA2 is steady
EE62 0A 2280 ASL A
EE63 10F5 2281 BPL BEE5A ;loop until clock bit rises
EE65 66A4 2282 ROR ZA4 ;add data bit to byte being read
EE67 AD00DD 2283 BEE67 LDA XDD00
EE6A CD00DD 2284 CMP XDD00
EE6D D0F8 2285 BNE BEE67 ;wait until PA2 is steady
EE6F 0A 2286 ASL A
EE70 30F5 2287 BMI BEE67 ;and for clock line to fall
EE72 C6A5 2288 DEC ZA5
EE74 D0E4 2289 BNE BEE5A ;loop until 8 bits read
EE76 20A0EE 2290 JSR SEEA0 ;set serial data high
EE79 2490 2291 BIT Z90
EE7B 5003 2292 BVC BEE80 ;if EOI is high
EE7D 2006EE 2293 JSR SEE06 ;pause 52 usec, set clock and data low
EE80 A5A4 2294 BEE80 LDA ZA4 ;restore byte read
EE82 58 2295 CLI ;enable IRQ
EE83 18 2296 CLC
EE84 60 2297 RTS

```

```

                2299 ;set serial clock line low
                2300 ;
EE85 AD00DD 2301 SEE85 LDA XDD00
EE88 29EF 2302     AND $EF
EE8A 8D00DD 2303     STA XDD00
EE8D 60 2304     RTS
                2305 ;
                2306 ;set serial clock line high
                2307 ;
EE8E AD00DD 2308 SEE8E LDA XDD00
EE91 0910 2309     ORA $10
EE93 8D00DD 2310     STA XDD00
EE96 60 2311     RTS
                2312 ;
                2313 ;set serial data line low
                2314 ;
EE97 AD00DD 2315 SEE97 LDA XDD00
EE9A 29DF 2316     AND $DF
EE9C 8D00DD 2317     STA XDD00
EE9F 60 2318     RTS
                2319 ;
                2320 ;set serial data line high
                2321 ;
EEA0 AD00DD 2322 SEEA0 LDA XDD00
EEA3 0920 2323     ORA $20
EEA5 8D00DD 2324     STA XDD00
EEA8 60 2325     RTS
                2326 ;
                2327 ;wait for PA2 to steady
                2328 ;exit with data bit in C
                2329 ;and clock bit in N
                2330 ;
EEA9 AD00DD 2331 SEEA9 LDA XDD00
EEAC CD00DD 2332     CMP XDD00
EEAF D0F8 2333     BNE SEEA9 ;wait until PA2 is steady
EEB1 0A 2334     ASL A ;shift data into C and clock into N
EEB2 60 2335     RTS
                2336 ;
                2337 ;delay 1 millisecond
                2338 ;
EEB3 8A 2339 SEEB3 TXA ;save A
EEB4 A2B8 2340 LDX $B8 ;initialize delay counter
EEB6 CA 2341 BEEB6 DEX
EEB7 D0FD 2342     BNE BEEB6 ;perform delay of 1 millisecond
EEB9 AA 2343     TAX ;restore A
EEBA 60 2344     RTS

```

```

2346 ;set next bit to transmit on RS-232 bus
2347 ;
EEBB A5B4 2348 SEEBB LDA ZB4 ;check pending bit count
EEBD F047 2349 BEQ BEF06 ;if byte finished, send stop bit(s)
EEBF 303F 2350 BMI BEF00 ;if not stop bit,
EEC1 46B6 2351 LSR ZB6 ;shift next bit to send into C
EEC3 A200 2352 LDX $00
EEC5 9001 2353 BCC BEEC8 ;X=00/FF for 0/1 to send
EEC7 CA 2354 DEX
EEC8 8A 2355 BEEC8 TXA
EEC9 45BD 2356 EOR ZBD ;compute parity of word being sent
EECB 85BD 2357 STA ZBD
EECD C6B4 2358 DEC ZB4 ;decrement bit count
EECF F006 2359 BEQ BEED7 ;if word finished, handle parity
EED1 8A 2360 BEED1 TXA
EED2 2904 2361 AND $04
EED4 85B5 2362 STA ZB5 ;set next bit to send
EED6 60 2363 RTS
2364 ;
EED7 A920 2365 BEED7 LDA $20 ;check RS-232 Command Register
EED9 2C9402 2366 BIT X0294 ;for parity option
EEDC F014 2367 BEQ BEEF2 ;no parity
EEDF 301C 2368 BMI BEEFC ;transmit mark or space parity
EEE0 7014 2369 BVS BEEF6 ;transmit even parity
EEE2 A5BD 2370 LDA ZBD ;make parity odd
EEE4 D001 2371 BNE BEEE7
EEE6 CA 2372 BEEE6 DEX
EEE7 C6B4 2373 BEEE7 DEC ZB4 ;decrement bit count
EEE9 AD9302 2374 LDA X0293 ;check RS-232 Control Register
EEEC 10E3 2375 BPL BEED1 ;and branch if only one stop bit needed
EEEE C6B4 2376 DEC ZB4 ;else decrement bit count
EEF0 D0DF 2377 BNE BEED1 ;and send first of 2 stop bits
EEF2 E6B4 2378 BEEF2 INC ZB4 ;increment bit count
EEF4 D0F0 2379 BNE BEEE6
EEF6 A5BD 2380 BEEF6 LDA ZBD ;use computed parity
EEF8 F0ED 2381 BEQ BEEE7 ;to set X for even parity
EEFA D0EA 2382 BNE BEEE6
EEFC 70E9 2383 BEEFC BVS BEEE7 ;transmit space parity
EEFE 50E6 2384 BVC BEEE6 ;transmit mark parity
EF00 E6B4 2385 BEF00 INC ZB4 ;bump bit count
EF02 A2FF 2386 LDX $FF
EF04 D0CB 2387 BNE BEED1 ;go send a stop bit
EF06 AD9402 2388 BEF06 LDA X0294 ;check RS-232 Command Register
EF09 4A 2389 LSR A
EF0A 9007 2390 BCC BEF13 ;and skip if in 3-line mode
EF0C 2C01DD 2391 BIT XDD01
EF0F 101D 2392 BPL BEF2E ;error if DSR signal missing
EF11 501E 2393 BVC BEF31 ;or CTS signal missing
EF13 A900 2394 BEF13 LDA $00
EF15 85BD 2395 STA ZBD ;clear parity
EF17 85B5 2396 STA ZB5 ;clear next bit to send
EF19 AE9802 2397 LDX X0298 ;move number of bits to send
EF1C 86B4 2398 STX ZB4 ;to bit count
EF1E AC9D02 2399 LDY X029D ;if RS-232 transmit buffer is not empty
EF21 CC9E02 2400 CPY X029E
EF24 F013 2401 BEQ BEF39
EF26 B1F9 2402 LDA (ZF9),Y ;move next byte to send
EF28 85B6 2403 STA ZB6 ;to character buffer
EF2A EE9D02 2404 INC X029D ;bump RS-232 transmit buffer output ptr
EF2D 60 2405 RTS

```

```

2407 ;handle RS-232 errors
2408 ;
EF2E A940 2409 BEF2E LDA $40 ;add DSR signal missing error bit
EF30 2C 2410 .BY $2C ;skip next instruction
EF31 A910 2411 BEF31 LDA $10 ;add CTS signal missing error bit
EF33 0D9702 2412 ORA X0297 ;to RS-232 Status Register
EF36 8D9702 2413 STA X0297
EF39 A901 2414 BEF39 LDA $01 ;clear TA2 IRQ mask
EF3B 8D0DDD 2415 JEF3B STA XDD0D ;set/clear ICR2 flags depending on A
EF3E 4DA102 2416 EOR X02A1
EF41 0980 2417 ORA $80
EF43 8DA102 2418 STA X02A1 ;set ICR2 activity register
EF46 8D0DDD 2419 STA XDD0D ;and ICR2
EF49 60 2420 RTS
2421 ;
2422 ;check control register to set word length
2423 ;
EF4A A209 2424 SEF4A LDX $09 ;start with a length of 9
EF4C A920 2425 LDA $20
EF4E 2C9302 2426 .BIT X0293 ;check RS-232 Control Register
EF51 F001 2427 BEQ BEF54 ;and branch if word length is 6 or 8
EF53 CA 2428 DEX ;else set X to 8
EF54 5002 2429 BEF54 BVC BEF58 ;if word length is 7 or 8, exit
EF56 CA 2430 DEX
EF57 CA 2431 DEX
EF58 60 2432 BEF58 RTS ;exit with length in X
2433 ;
2434 ;add bit input on RS-232 bus to word being input
2435 ;
EF59 A6A9 2436 JEF59 LDX ZA9 ;check receiver start bit flag
EF5B D033 2437 BNE BEF90 ;if a bit is ready,
EF5D C6A8 2438 DEC ZA8 ;decrement input bit count
EF5F F036 2439 BEQ BEF97 ;if word is not complete,
EF61 300D 2440 BMI BEF70
EF63 A5A7 2441 LDA ZA7 ;use receiver input bit
EF65 45AB 2442 EOR ZAB ;to calculate parity
EF67 85AB 2443 STA ZAB
EF69 46A7 2444 LSR ZA7 ;shift bit received
EF6B 66AA 2445 ROR ZAA ;into byte being read
EF6D 60 2446 BEF6D RTS
2447 ;
2448 ;handle end of word for RS-232 input
2449 ;
EF6E C6A8 2450 BEF6E DEC ZA8 ;decrement input bit count
EF70 A5A7 2451 BEF70 LDA ZA7 ;check receiver input bit
EF72 F067 2452 BEQ BEFDB ;if something read
EF74 AD9302 2453 LDA X0293 ;use RS-232 Control Register to
EF77 0A 2454 ASL A ;shift stop bit count flag into C
EF78 A901 2455 LDA $01
EF7A 65A8 2456 ADC ZA8 ;add possible second stop bit
EF7C D0EF 2457 BNE BEF6D ;to input bit count and exit

```

```

2459 ;enable byte reception
2460 ;
EF7E A990 2461 BEF7E LDA $90
EF80 8D0DD 2462 STA XDD0D ;set Flag bit mask in ICR2
EF83 0DA102 2463 ORA X02A1 ;plus ICR2 activity register
EF86 8DA102 2464 STA X02A1 ;to form new ICR2 activity register
EF89 85A9 2465 STA ZA9 ;and receiver start bit flag
EF8B A902 2466 LDA $02
EF8D 4C3BEF 2467 JMP JEF3B ;set TB2 mask in ICR2 and exit
2468 ;
2469 ;receiver start bit test
2470 ;
EF90 A5A7 2471 BEF90 LDA ZA7 ;test receiver input bit
EF92 D0EA 2472 BNE BEF7E ;if start read,
EF94 85A9 2473 STA ZA9 ;clear start bit flag
EF96 60 2474 RTS
2475 ;
2476 ;put received data into RS-232 receive buffer
2477 ;
EF97 AC9B02 2478 BEF97 LDY X029B ;if RS-232 receive buffer input pointer
EF9A C8 2479 INY
EF9B CC9C02 2480 CPY X029C ;= RS-232 receive buffer output pointer
EF9E F02A 2481 BEQ BEFCA ;then receive buffer overrun error
EFA0 8C9B02 2482 STY X029B ;bump RS-232 receive buffer input ptr
EFA3 88 2483 DEY
EFA4 A5AA 2484 LDA ZAA ;fetch byte received so far in A
EFA6 AE9802 2485 LDY X0298 ;and number of bits to receive in X
EFA9 E009 2486 BEFA9 CPX $09
EFAB F004 2487 BEQ BEFB1
EFAD 4A 2488 LSR A ;right justify input byte
EFAE E8 2489 INX
EFAF D0F8 2490 BNE BEFA9
EFB1 91F7 2491 BEFB1 STA (ZF7),Y ;store received byte in receive buffer
EFB3 A920 2492 LDA $20 ;check RS-232 Command Register
EFB5 2C9402 2493 BIT X0294 ;for parity options
EFB8 F0B4 2494 BEQ BEF6E ;no parity
EFBA 30B1 2495 BMI BEF6D ;parity check disabled
EFBC A5A7 2496 LDA ZA7 ;fetch receiver parity bit
EFBE 45AB 2497 EOR ZAB ;check against calculated parity
EFC0 F003 2498 BEQ BEFC5 ;odd parity received
EFC2 70A9 2499 BVS BEF6D ;exit if even parity required
EFC4 2C 2500 .BY $2C ;skip next instruction
EFC5 50A6 2501 BEFC5 BVC BEF6D ;exit if odd parity required
EFC7 A901 2502 LDA $01 ;else indicate Parity Error
EFC9 2C 2503 .BY $2C ;skip next instruction
EFCA A904 2504 BEFCA LDA $04 ;indicate Receiver Buffer Overrun error
EFCB 2C 2505 .BY $2C ;skip next instruction
EFCD A980 2506 BEFCD LDA $80 ;indicate Break Detected error
EFCF 2:1 2507 .BY $2C ;skip next instruction
EFD0 A902 2508 BEFDD LDA $02 ;Framing error
EFD2 0D9702 2509 ORA X0297 ;plus original RS-232 Status Register
EFD5 8D9702 2510 STA X0297 ;makes new RS-232 Status Register
EFD8 4C7EEF 2511 JMP BEF7E ;enable byte reception and exit
2512 ;
EFDB A5AA 2513 BEFDB LDA ZAA ;if byte received so far is non-zero
EFD D0F1 2514 BNE BEFDD ;indicate framing error
EFD F0EC 2515 BEQ BEFCD ;or break detected error

```

```

2517 ;output on RS-232 device
2518 ;
EFE1 859A 2519 JEFEL STA Z9A ;set output device
EFE3 AD9402 2520 LDA X0294 ;check RS-232 Command Register
EFE6 4A 2521 LSR A ;for 3-line or X-line mode
EFE7 9029 2522 BCC BF012 ;and skip if 3-line
EFE9 A902 2523 LDA $02
EFEB 2C01DD 2524 BIT XDD01
EFEE 101D 2525 BPL BF00D ;indicate error if DSR signal missing
EFF0 D020 2526 BNE BF012 ;exit if RTS present
EFF2 ADA102 2527 BEFF2 LDA X02A1 ;check ICR2 activity register
EFF5 2902 2528 AND $02
EFF7 D0F9 2529 BNE BEFF2 ;and loop until byte received
EFF9 2C01DD 2530 BEFF9 BIT XDD01
EFFC 70FB 2531 BVS BEFF9 ;wait for CTS signal
EFPE AD01DD 2532 LDA XDD01
F001 0902 2533 ORA $02
F003 814321 2534 STA XDD01 ;set RTS signal
F006 2C01DD 2535 BF006 BIT XDD01
F009 7007 2536 BVS BF012 ;exit if CTS is high
F00B 30F9 2537 BMI BF006 ;loop if DSR is high
F00D A940 2538 BF00D LDA $40 ;set DSR Signal Missing Error
F00F 8D9702 2539 STA X0297 ;in RS-232 Status Register
F012 18 2540 BF012 CLC
F013 60 2541 RTS
2542 ;
2543 ;buffer character to output on RS-232
2544 ;
F014 2028F0 2545 BF014 JSR SF028 ;schedule TA2 if not transmitting
F017 AC9E02 2546 SF017 LDY X029E
F01A C8 2547 INY ;if no room in the RS-232 buffer
F01B CC9D02 2548 CPY X029D
F01E F0F4 2549 BEQ BF014 ;wait until there is room
F020 8C9E02 2550 STY X029E ;bump RS-232 transmit buffer output ptr
F023 88 2551 DEY
F024 A59E 2552 LDA Z9E ;move, buffered character
F026 91F9 2553 STA (ZF9),Y ;to transmit buffer
F028 ADA102 2554 SF028 LDA X02A1 ;check ICR2 activity register
F02B 4A 2555 LSR A
F02C B01E 2556 BCS BF04C ;if not transmitting,
F02E A910 2557 LDA $10
F030 8D0EDD 2558 STA XDD0E ;force load into CRA2
F033 AD9902 2559 LDA X0299 ;move baud rate full bit time
F036 8D04DD 2560 STA XDD04 ;into TAL2
F039 AD9A02 2561 LDA X029A
F03C 8D05DD 2562 STA XDD05 ;and TAB2
F03F A981 2563 LDA $81
F041 203BEF 2564 JSR JEF3B ;set TA2 mask bit in ICR2
F044 2006EF 2565 JSR BEF06 ;perform initialization for new byte
F047 A911 2566 LDA $11 ;set force load and start TA2 in CRA2
F049 8D0EDD 2567 STA XDD0E
F04C 60 2568 BF04C RTS

```



```

2570 ;initialize RS-232 input
2571 ;
F04D 8599 2572 JF04D STA Z99 ;set input device
F04F AD9402 2573 LDA X0294 ;check RS-232 Command Register
F052 4A 2574 LSR A ;for 3-line or X-line mode
F053 9028 2575 BCC BF07D ;and skip if 3-line
F055 2908 2576 AND $08 ;check for full/half duplex mode
F057 F024 2577 BEQ BF07D ;and skip if full duplex
F059 A902 2578 LDA $02
F05B 2C01DD 2579 BIT XDD01
F05E 10AD 2580 BPL BF00D ;error if DSR signal not present
F060 F022 2581 BEQ BF084
F062 ADA102 2582 BF062 LDA X02A1 ;check ICR2 activity register
F065 4A 2583 LSR A
F066 B0FA 2584 BCS BF062 ;if not transmitting,
F068 AD01DD 2585 LDA XDD01
F06B 29FD 2586 AND $FD ;clear RTS line
F06D 8D01DD 2587 STA XDD01
F070 AD01DD 2588 BF070 LDA XDD01
F073 2904 2589 AND $04
F075 F0F9 2590 BEQ BF070 ;wait for DSR
F077 A990 2591 BF077 LDA $90
F079 18 2592 CLC
F07A 4C3BEF 2593 JMP JEF3B ;set Flag bit mask in ICR2 and return
2594 ;
F07D ADA102 2595 BF07D LDA X02A1 ;fetch ICR2 activity register
F080 2912 2596 AND $12 ;if receiving/waiting for receiver edge,
F082 F0F3 2597 BEQ BF077 ;set Flag mask
F084 18 2598 BF084 CLC
F085 60 2599 RTS
2600 ;
2601 ;get next character from RS-232 input buffer
2602 ;
F086 AD9702 2603 SF086 LDA X0297 ;A = contents of RS-232 Status Register
F089 AC9C02 2604 LDY X029C
F08C CC9B02 2605 CPY X029B ;if receiver buffer is empty,
F08F F00B 2606 BEQ BF09C ;set flag in RS-232 status register
F091 29F7 2607 AND $F7 ;else clear Receiver Buffer Empty flag
F093 8D9702 2608 STA X0297 ;in RS-232 Status Register
F096 B1F7 2609 LDA (2F7),Y ;fetch character from buffer
F098 EE9C02 2610 INC X029C ;bump RS-232 receive buffer output ptr
F09B 60 2611 RTS
2612 ;
F09C 0908 2613 BF09C ORA $08 ;indicate Receiver Buffer Empty
F09E 8D9702 2614 STA X0297 ;in RS-232 Status Register
FOA1 A900 2615 LDA $00 ;and return with a null character
FOA3 60 2616 RTS

```

```

2618 ;protect serial/cassette routine from RS-232 NMI's
2619 ;
FOA4 48      2620 SFOA4 PHA          ;save A
FOA5 ADA102  2621 LDA X02A1        ;if no activity in ICR2
FOA8 F011    2622 BEQ BFOBB        ;exit
FOAA ADA102  2623 BFOAA LDA X02A1
FOAD 2903    2624 AND $03
FOAF D0F9    2625 BNE BFOAA        ;else wait until not sending/receiving
FOB1 A910    2626 LDA $10
FOB3 8D0DDD  2627 STA XDDOD        ;clear Flag bit mask in ICR2
FOB6 A900    2628 LDA $00
FOB8 8DA102  2629 STA X02A1        ;clear ICR2 activity register
FOBB 68      2630 BFOBB PLA        ;restore A
FOBC 60      2631 RTS
2632 ;
2633 ;Kernal I/O messages
2634 ;
FOBD OD492F  2635 TFOBD .BY $OD,"I","O","E","R","R","O","R"," "+$80
FOC9 OD5345  2636 TFOC9 .BY $OD,"S","E","A","R","C","H","I","N","G"," "+$80
FOD4 464F52  2637 TFOD4 .BY "F","O","R"," "+$80
FOD8 OD5052  2638 TFOD8 .BY $OD,"P","R","E","S","S","P","L","A","Y","O","N","
FOE7 544150  2639 .BY "I","A","P","E"+$80
FOEB 505245  2640 TFOEB .BY "P","R","E","S","S","R","E","C","O","R","D","&."
FOFA 504C41  2641 .BY "P","L","A","Y","O","N","T","A","P","E"+$80
FI06 OD4C4F  2642 TF106 .BY $OD,"L","O","A","D","I","N","G"+$80
FI0E OD5341  2643 TF10E .BY $OD,"S","A","V","I","N","G"," "+$80
FI16 OD5645  2644 TF116 .BY $OD,"V","E","R","I","F","Y","I","N","G"+$80
FI20 OD464F  2645 TF120 .BY $OD,"F","O","U","N","D"," "+$80
FI27 OD4F4B  2646 TF127 .BY $OD,"O","K,$8D
2647 ;
2648 ;print kernal message indexed by Y
2649 ;
FI2B 249D    2650 JFI2B BIT Z9D      ;exit if not in direct mode
FI2D 100D    2651 BPL BF13C
FI2F B9BDFO  2652 BF12F LDA TFOBD,Y    ;fetch a character from message table
FI32 08      2653 PHF
FI33 297F    2654 AND $7F          ;strip bit 7
FI35 20D2FF  2655 JSR SFFD2        ;print character
FI38 C8      2656 INY            ;advance index
FI39 28      2657 PLP            ;if last char did not contain bit 7,
FI3A 10F3    2658 BPL BF12F        ;repeat
FI3C 18      2659 BF13C CLC
FI3D 60      2660 RTS
2661 ;
2662 ;get a character
2663 ;
FI3E A599    2664 WFI3E LDA Z99        ;if input device
FI40 D008    2665 BNE BF14A        ;is the keyboard
FI42 A5C6    2666 LDA ZC6          ;and keyboard buffer
FI44 F00F    2667 BEQ BF155        ;is not empty
FI46 78      2668 SEI            ;disable IRQ
FI47 4CB4E5  2669 JMP SE5B4        ;then fetch char from keyboard queue
2670 ;
FI4A C902    2671 BF14A CMP $02      ;if input device is RS-232
FI4C D018    2672 BNE BF166
FI4E 8497    2673 SFI4E STY Z97      ;save Y
FI50 2086FO  2674 JSR SF086        ;get a char from RS-232 input buffer
FI53 A497    2675 LDY Z97          ;restore Y
FI55 18      2676 BF155 CLC
FI56 60      2677 RTS

```

	2679	;input a character	
	2680	;	
F157 A599	2681	WF157 LDA Z99	;if the input device
F159 D008	2682	BNE BF166	;is the keyboard
F15B ASD3	2683	LDA ZD3	;move position of cursor on line
F15D 85CA	2684	STA ZCA	;to input cursor position on line
F15F ASD6	2685	LDA ZD6	;move cursor line number
F161 85C9	2686	STA ZC9	;to input cursor line number
F163 4C32E6	2687	JMP JE632	;and go input from the keyboard
	2688	;	
F166 C903	2689	BF166 CMP \$03	;if input device is the screen
F168 D009	2690	BNE BF173	
F16A 85D0	2691	STA ZD0	;set screen/keyboard flag to screen
F16C ASD5	2692	LDA ZD5	;move screen line length
F16E 85C8	2693	STA ZC8	;to end of line pointer
F170 4C32E6	2694	JMP JE632	;and go input from the screen
	2695	;	
F173 B038	2696	BF173 BCS BF1AD	;if device > 3 (serial bus), handle
F175 C902	2697	CMP \$02	
F177 F03F	2698	BEQ BF1B8	;if input device is RS-232, handle
F179 8697	2699	STX Z97	;input device is tape, save X
F17B 2099F1	2700	JSR SF199	;fetch a byte from tape buffer
F17E B016	2701	BCS BF196	;exit upon error
F180 48	2702	PHA	;save byte on stack
F181 2099F1	2703	JSR SF199	;look ahead in tape buffer
F184 B00D	2704	BCS BF193	;and exit upon error
F186 D005	2705	BNE BF18D	;if at end of file,
F188 A940	2706	LDA \$40	;set end of file bit
F18A 201CFE	2707	JSR SFE1C	;in ST
F18D C6A6	2708	BF18D DEC ZA6	;adjust tape buffer ptr back to normal
F18F A697	2709	LDX Z97	;restore X
F191 68	2710	PLA	;restore byte read
F192 60	2711	RTS	
	2712	;	
F193 AA	2713	BF193 TAX	
F194 68	2714	PLA	;set stack straight
F195 8A	2715	TXA	
F196 A697	2716	BF196 LDX Z97	;restore X
F198 60	2717	RTS	;error exit with C set

```

2719 ;read a byte from the cassette buffer
2720 ;
F199 200DF8 2721 SF199 JSR SF80D      ;add 1 to buffer pointer
F19C D00B 2722      BNE BF1A9      ;if end of buffer,
F19E 2041F8 2723      JSR SF841      ;read next block
F1A1 B011 2724      BCS BF1B4      ;and exit upon error
F1A3 A900 2725      LDA $00
F1A5 85A6 2726      STA ZA6        ;reset tape buffer pointer
F1A7 F0F0 2727      BEQ SF199      ;and loop back
F1A9 B1B2 2728 BF1A9 LDA (ZB2),Y    ;fetch byte from tape buffer
F1AB 18 2729      CLC              ;indicate no error
F1AC 60 2730      RTS              ;and return
2731 ;
F1AD A590 2732 BF1AD LDA Z90        ;if ST indicates Device Not Present
F1AF F004 2733      BEQ BF1B5
F1B1 A90D 2734 BF1B1 LDA $0D        ;force a return
F1B3 18 2735 BF1B3 CLC
F1B4 60 2736 BF1B4 RTS
2737 ;
2738 ;read a byte from the serial bus
2739 ;
F1B5 4C13EE 2740 BF1B5 JMP JEE13     ;input byte from serial bus and return
2741 ;
2742 ;read a byte from the RS-232 bus
2743 ;
F1B8 204EF1 2744 BF1B8 JSR SF14E     ;get a char from RS-232 input buffer
F1BB B0F7 2745      BCS BF1B4      ;exit upon error
F1BD C900 2746      CMP $00
F1BF D0F2 2747      BNE BF1B3      ;exit if receive buffer not empty
F1C1 AD9702 2748      LDA X0297     ;if RS-232 Status Register
F1C4 2960 2749      AND $60        ;indicates DSR Missing,
F1C6 D0E9 2750      BNE BF1B1      ;force a Return
F1C8 F0EE 2751      BEQ BF1B8      ;else repeat

```

	2753	;output a character	
	2754	;	
F1CA 48	2755	WF1CA PHA	;save character on stack
F1CB A59A	2756	LDA Z9A	;if output device
F1CD C903	2757	CMP \$03	;is the screen
F1CF D004	2758	BNE BF1D5	
F1D1 68	2759	PLA	;restore character
F1D2 4C16E7	2760	JMP SE716	;and output to screen
	2761	;	
F1D5 9004	2762	BF1D5 BCC BF1DB	;if output device > 3
F1D7 68	2763	PLA	;restore character
F1D8 4CDDDE	2764	JMP JEDDD	;and output to serial bus
	2765	;	
F1DB 4A	2766	BF1DB LSR A	;set C to 1 for cassette
F1DC 68	2767	PLA	;move character to output
F1DD 859E	2768	SF1DD STA Z9E	;to temporary storage
F1DF 8A	2769	TXA	
F1E0 48	2770	PHA	
F1E1 98	2771	TYA	
F1E2 48	2772	PHA	;save XY on stack
F1E3 9023	2773	BCC BF208	;if output device is cassette
F1E5 200DF8	2774	JSR SF80D	;add one to tape buffer index
F1E8 D00E	2775	BNE BF1F8	;if buffer is full, (192 bytes)
F1EA 2064F8	2776	JSR SF864	;write block to cassette
F1ED 800E	2777	BCS BF1FD	;and exit upon error
F1EF A902	2778	LDA \$02	;use use Data code
F1F1 A000	2779	LDY \$00	
F1F3 91B2	2780	STA (ZB2),Y	;in beginning of next tape buffer
F1F5 C8	2781	INY	
F1F6 84A6	2782	STY ZA6	;initialize tape buffer pointer
F1F8 A59E	2783	BF1F8 LDA Z9E	;move character to output
F1FA 91B2	2784	STA (ZB2),Y	;into the tape buffer
F1FC 18	2785	JF1FC CLC	;clear error flag
F1FD 68	2786	BF1FD PLA	
F1FE A8	2787	TAY	
F1FF 68	2788	PLA	
F200 AA	2789	TAX	;restore XY
F201 A59E	2790	LDA Z9E	;restore character just output
F203 9002	2791	BCC BF207	;exit if no errors
F205 A900	2792	LDA \$00	;else return with A=0
F207 60	2793	BF207 RTS	
	2794	;	
F208 2017F0	2795	BF208 JSR SF017	;out a character to RS-232 channel
F20B 4CFCF1	2796	JMP JF1FC	;and exit

```

2798 ;set input device
2799 ;
F20E 200FF3 2800 WF20E JSR SF30F ;check X against logical file table
F211 F003 2801 BEQ BF216 ;if not matched,
F213 4C01F7 2802 JMP JF701 ;indicate File Not Open
2803 ;
F216 201FF3 2804 BF216 JSR SF31F ;set logical file, device and sec. adr
F219 A5BA 2805 LDA ZBA ;if device is keyboard,
F21B F016 2806 BEQ BF233 ;exit
F21D C903 2807 CMP $03 ;if device is screen,
F21F F012 2808 BEQ BF233 ;exit
F221 B014 2809 BCS BF237 ;handle if device is serial bus (> 3)
F223 C902 2810 CMP $02
F225 D003 2811 BNE BF22A ;if device is RS-232,
F227 4C4DF0 2812 JMP JF04D ;initialize RS-232 input
2813 ;
F22A A6B9 2814 BF22A LDX ZB9 ;if input device is tape and sec. adr
F22C E060 2815 CPX $60 ;indicates that this is an input file
F22E F003 2816 BEQ BF233 ;great,
F230 4C0AF7 2817 JMP JF70A ;else indicate Not Input File Error
2818 ;
F233 8599 2819 BF233 STA Z99 ;set input device
F235 18 2820 CLC ;clear error flag
F236 60 2821 RTS
2822 ;
2823 ;set serial bus input device
2824 ;
F237 AA 2825 BF237 TAX
F238 2009ED 2826 JSR SED09 ;send TALK on serial bus
F23B A5B9 2827 LDA ZB9 ;if secondary address specified,
F23D 1006 2828 BPL BF245 ;send it
F23F 20CCED 2829 JSR SEDCC ;else send dummy secondary address
F242 4C48F2 2830 JMP JF248
2831 ;
F245 20C7ED 2832 BF245 JSR SEDC7 ;send secondard addr (after TALK)
F248 8A 2833 JF248 TXA
F249 2490 2834 BIT Z90
F24B 10E6 2835 BPL BF233 ;if ST shows device present, exit
F24D 4C07F7 2836 JMP JF707 ;else indicate error

```

	2838	;set output device	
	2839	;	
F250 200FF3	2840	WF250 JSR SF30F	;check X against logical file table
F253 F003	2841	BEQ BF258	
F255 4C01F7	2842	JMP JF701	;File Not Open Error if not found
	2843	;	
F258 201FF3	2844	BF258 JSR SF31F	;set logical file, device and sec. adr
F25B A5BA	2845	LDA ZBA	;if current device
F25D D003	2846	BNE BF262	;indicates an input file
F25F 4C0DF7	2847	BF25F JMP JF70D	;error Not Output File
	2848	;	
F262 C903	2849	BF262 CMP \$03	;if device is the screen
F264 F00F	2850	BEQ BF275	;exit
F266 B011	2851	BCS BF279	;if serial bus device, handle
F268 C902	2852	CMF \$02	;if device is RS-232
F26A D003	2853	BNE BF26F	
F26C 4CE1EF	2854	JMP JEFE1	;initialize RS-232 output
	2855	;	
F26F A6B9	2856	BF26F LDX ZB9	;if secondary address
F271 E060	2857	CPX \$60	;indicates an input file
F273 F0EA	2858	BEQ BF25F	;Not Output File Error
F275 859A	2859	BF275 STA Z9A	Xstore output device
F277 18	2860	CLC	;clear error indication
F278 60	2861	RTS	
	2862	;	
	2863	;set serial bus output device	
	2864	;	
F279 AA	2865	BF279 TAX	
F27A 200CED	2866	JSR SEDOC	;send LISTEN on serial bus
F27D A5B9	2867	LDA ZB9	;if secondary address specified,
F27F 1005	2868	BPL BF286	;send it
F281 20BEED	2869	JSR SEDBE	;else send dummy secondary address
F284 D003	2870	BNE BF289	;skip next instruction
F286 20B9ED	2871	BF286 JSR SEDEB9	;send secondary addr after LISTEN
F289 8A	2872	BF289 TXA	
F28A 2490	2873	BIT Z90	;check ST
F28C 10E7	2874	BPL BF275	;and set output device if no errors
F28E 4C07F7	2875	JMP JF707	;else indicate Device Not Present Error

	2877	;CLOSE a file	
	2878	;	
F291 2014F3	2879	WF291 JSR SF314	;check A against logical file table
F294 F002	2880	BEQ BF298	;continue if found
F296 18	2881	CLC	;else exit
F297 60	2882	RTS	
	2883	;	
F298 201FF3	2884	BF298 JSR SF31F	;set logical file, device and sec. adr
F29B 8A	2885	TXA	
F29C 48	2886	PHA	;save file index
F29D A5BA	2887	LDA ZBA	;if current device is keyboard
F29F F050	2888	BEQ BF2F1	
F2A1 C903	2889	CMP \$03	
F2A3 F04C	2890	BEQ BF2F1	;or screen, skip to end
F2A5 B047	2891	BCS BF2EE	;if serial bus device (> 3), handle
F2A7 C902	2892	CMP \$02	
F2A9 D01D	2893	BNE BF2C8	;if RS-232 device,
F2AB 68	2894	PLA	;restore file index
F2AC 20F2F2	2895	JSR SF2F2	;re-organize file tables
F2AF 2083F4	2896	JSR SF483	;re-initialize CIA2
F2B2 2027FE	2897	JSR SFE27	;read top of memory into XY
F2B5 A5F8	2898	LDA ZF8	;if high byte of RS-232 receive buffer
F2B7 F001	2899	BEQ BF2BA	;base address is not 0,
F2B9 C8	2900	INY	;add 1 to high byte of top of memory
F2BA A5FA	2901	BF2BA LDA ZFA	;if high byte of RS-232 transmit buffer
F2BC F001	2902	BEQ BF2BF	;base address is not 0
F2BE C8	2903	INY	;add 1 to high byte of top of memory
F2BF A900	2904	BF2BF LDA \$00	;clear high byte of RS-232
F2C1 85F8	2905	STA ZF8	;receive and transmit buffer
F2C3 85FA	2906	STA ZFA	;base addresses
F2C5 4C7DF4	2907	JMP JF47D	;set new top of memory from XY


```

2909 ;close cassette device
2910 ;
F2C8 A5B9 2911 BF2C8 LDA ZB9 ;if secondary address
F2CA 290F 2912 AND $0F ;indicates an input file,
F2CC F023 2913 BEQ BF2F1 ;skip following
F2CE 20D0F7 2914 JSR SF7D0 ;else get tape buffer pointer in XY
F2D1 A900 2915 LDA $00
F2D3 38 2916 SEC
F2D4 20DDF1 2917 JSR SF1DD ;close file with a $00
F2D7 2064F8 2918 JSR SF864 ;write last block to cassette
F2DA 9004 2919 BCC BF2E0
F2DC 68 2920 PLA ;and return with C set upon error
F2DD A900 2921 LDA $00
F2DF 60 2922 RTS
2923 ;
F2E0 A5B9 2924 BF2E0 LDA ZB9 ;if secondary address indicates that
F2E2 C962 2925 CMP $62 ;EOT marker to be written,
F2E4 D00B 2926 BNE BF2F1
F2E6 A905 2927 LDA $05
F2E8 206AF7 2928 JSR SF76A ;write special block to tape with EOT
F2EB 4CF1F2 2929 JMP BF2F1 ;then re-organize file tables
2930 ;
2931 ;close serial bus device
2932 ;
F2EE 2042F6 2933 BF2EE JSR SF642 ;close serial bus device
F2F1 68 2934 BF2F1 PLA ;restore file index
2935 ;
2936 ;re-organize file tables
2937 ;
F2F2 AA 2938 SF2F2 TAX
F2F3 C698 2939 DEC Z98 ;decrement # files open
F2F5 E498 2940 CPX Z98
F2F7 F014 2941 BEQ BF30D ;exit if all files closed
F2F9 A498 2942 LDY Z98
F2FB B95902 2943 LDA X0259,Y ;else move parameters of last entry
F2FE 9D5902 2944 STA X0259,X ;(file, device and secondary address)
F301 B96302 2945 LDA X0263,Y ;to slot of file being closed
F304 9D6302 2946 STA X0263,X ;this removes file being closed
F307 B96D02 2947 LDA X026D,Y
F30A 9D6D02 2948 STA X026D,X
F30D 18 2949 BF30D CLC
F30E 60 2950 RTS

```

```

2952 ;check X against logical file table
2953 ;
F30F A900 2954 SF30F LDA $00
F311 8590 2955 STA Z90 ;clear status word ST
F313 8A 2956 TXA
F314 A698 2957 SF314 LDX Z98 ;use # of files open to begin search
F316 CA 2958 BF316 DEX
F317 3015 2959 BMI BF32E ;exit if file not found
F319 DD5902 2960 CMP X0259,X
F31C D0F8 2961 BNE BF316 ;repeat if not matched
F31E 60 2962 RTS
2963 ;
2964 ;set file parameters depending on X
2965 ;
F31F BD5902 2966 SF31F LDA X0259,X
F322 85B8 2967 STA ZB8 ;set current logical file
F324 BD6302 2968 LDA X0263,X
F327 85BA 2969 STA ZBA ;set current device
F329 BD6D02 2970 LDA X026D,X
F32C 85B9 2971 STA ZB9 ;set current secondary address
F32E 60 2972 BF32E RTS
2973 ;
2974 ;close all files
2975 ;
F32F A900 2976 WF32F LDA $00
F331 8598 2977 STA Z98 ;clear number of files open
2978 ;
2979 ;restore I/O to default devices
2980 ;
F333 A203 2981 WF333 LDX $03
F335 E49A 2982 CPX Z9A ;if output device is not the serial bus
F337 B003 2983 BCS BF33C
F339 20FEED 2984 JSR SEDFE ;send UNLISTEN on serial bus
F33C E499 2985 BF33C CPX Z99 ;if input device is not on serial bus,
F33E B003 2986 BCS BF343
F340 20EFED 2987 JSR SEDEF ;send TALK on serial bus
F343 869A 2988 BF343 STX Z9A ;set output device to screen
F345 A900 2989 LDA $00
F347 8599 2990 STA Z99 ;set input device to keyboard
F349 60 2991 RTS

```

```

2993 ;OPEN a file
2994 ;
F34A A6B8 2995 WF34A LDX ZB8 ;if current logical file is 0
F34C D003 2996 BNE BF351
F34E 4C0AF7 2997 JMP JF70A ;indicate Not Input File Error
2998 ;
F351 200FF3 2999 BF351 JSR SF30F ;check X against logical file table
F354 D003 3000 BNE BF359 ;if found,
F356 4CFEF6 3001 JMP JF6FE ;File Open Error
3002 ;
F359 A698 3003 BF359 LDX Z98 ;if # files open is 10 or more,
F35B E00A 3004 CPX $0A
F35D 9003 3005 BCC BF362
F35F 4CFEF6 3006 JMP JF6FB ;Too Many Files Error
3007 ;
F362 E698 3008 BF362 INC Z98 ;add 1 to # files open
F364 A5B8 3009 LDA ZB8 ;add current logical file
F366 9D5902 3010 STA X0259,X ;to end of table of logical files open
F369 A5B9 3011 LDA ZB9 ;add secondary address
F36B 0960 3012 ORA $60 ;plus offset
F36D 85B9 3013 STA ZB9 ;to create current secondary address
F36F 9D6D02 3014 STA X026D,X ;and enter into secondary address table
F372 A5BA 3015 LDA ZBA ;add current device
F374 9D6302 3016 STA X0263,X ;to current device table
F377 F05A 3017 BEQ BF3D3 ;and exit if device is keyboard
F379 C903 3018 CMP $03
F37B F056 3019 BEQ BF3D3 ;or screen
F37D 9005 3020 BCC BF384
F37F 20D5F3 3021 JSR SF3D5 ;for serial device (> 3), perform open
F382 904F 3022 BCC BF3D3 ;JMP
F384 C902 3023 BF384 CMP $02
F386 D003 3024 BNE BF38B
F388 4C09F4 3025 JMP JF409 ;for RS-232 device, perform open
3026 ;
3027 ;open for cassette device
3028 ;
F38B 20D0F7 3029 BF38B JSR SF7D0 ;get tape buffer address in XY
F38E B003 3030 BCS BF393 ;if address too low,
F390 4C13F7 3031 JMP JF713 ;indicate Illegal Device #
3032 ;
F393 A5B9 3033 BF393 LDA ZB9 ;get current secondary address
F395 290F 3034 AND $0F ;mask low order 4 bits
F397 D01F 3035 BNE BF3B8 ;skip if open for output
F399 2017F8 3036 JSR SF817 ;handle msgs for cassette read
F39C B036 3037 BCS BF3D4 ;exit upon error
F39E 20AFF5 3038 JSR SF5AF ;handle messages for load
F3A1 A5B7 3039 LDA ZB7 ;get # characters in file name
F3A3 F00A 3040 BEQ BF3AF
F3A5 20EAF7 3041 JSR SF7EA ;if name present, search tape for file
F3A8 9018 3042 BCC BF3C2 ;if not matched
F3AA F028 3043 BEQ BF3D4 ;and not end of tape,
F3AC 4C04F7 3044 BF3AC JMP JF704 ;File Not Found Error
3045 ;
F3AF 202CF7 3046 BF3AF JSR SF72C ;since no name, get next tape data
F3B2 F020 3047 BEQ BF3D4 ;exit if end of tape
F3B4 900C 3048 BCC BF3C2 ;if no errors, continue
F3B6 B0F4 3049 BCS BF3AC ;else indicate File Not Found Error

```

```

3051 ;open cassette for output
3052 ;
F3B8 2038F8 3053 BF3B8 JSR SF838 ;handle msgs for cassette write
F3BB B017 3054 BCS BF3D4 ;exit upon error
F3BD A904 3055 LDA $04
F3BF 206AF7 3056 JSR SF76A ;write block with Data Header code
F3C2 A9BF 3057 BF3C2 LDA $BF ;initialize buffer index
F3C4 A4B9 3058 LDY ZB9 ;check secondary address
F3C6 C060 3059 CPY $60 ;if tape opened for output
F3C8 F007 3060 BEQ BF3D1
F3CA A000 3061 LDY $00
F3CC A902 3062 LDA $02 ;put code for Data Block
F3CE 91B2 3063 STA (ZB2),Y ;into first byte of buffer
F3D0 98 3064 TYA
F3D1 85A6 3065 BF3D1 STA ZA6 ;initialize tape buffer pointer
F3D3 18 3066 BF3D3 CLC
F3D4 60 3067 BF3D4 RTS
3068 ;
3069 ;open for serial bus devices (device > 3)
3070 ;
F3D5 A5B9 3071 SF3D5 LDA ZB9 ;if current secondary address invalid,
F3D7 30FA 3072 BMI BF3D3 ;exit
F3D9 A4B7 3073 LDY ZB7 ;if name not present,
F3DB F0F6 3074 BEQ BF3D3 ;exit
F3DD A900 3075 LDA $00
F3DF 8590 3076 STA Z90 ;else clear ST
F3E1 A5BA 3077 LDA ZBA ;set current device
F3E3 200CED 3078 JSR SED0C ;send UNLISTEN on serial bus
F3E6 A5B9 3079 LDA ZB9 ;send current secondary address
F3E8 09F0 3080 ORA $F0 ;plus offset
F3EA 20B9ED 3081 JSR SEDB9 ;on serial bus
F3ED A590 3082 LDA Z90 ;check ST
F3EF 1005 3083 BPL BF3F6 ;if Device Not Present,
F3F1 68 3084 PLA ;delete own return address
F3F2 68 3085 PLA
F3F3 4C07F7 3086 JMP JF707 ;indicate Device Not Present
3087 ;
F3F6 A5B7 3088 BF3F6 LDA ZB7 ;if file name present
F3F8 F00C 3089 BEQ BF406
F3FA A000 3090 LDY $00 ;initialize index
F3FC B1BB 3091 BF3FC LDA (ZBB),Y
F3FE 20DDED 3092 JSR JEDDD ;send file name on serial bus
F401 C8 3093 INY
F402 C4B7 3094 CPY ZB7
F404 D0F6 3095 BNE BF3FC ;repeat for all characters of file name.
F406 4C54F6 3096 BF406 JMP JF654 ;send UNLISTEN on serial bus and exit

```

```

3098 ;OPEN RS-232 device
3099 ;
F409 2083F4 3100 JF409 JSR SF483 ;initialize CIA2
F40C 8C9702 3101 STY X0297 ;clear RS-232 Status Register
F40F C4B7 3102 BF40F CPY ZB7 ;if pseudo file name present,
F411 F00A 3103 BEQ BF41D
F413 B1BB 3104 LDA (ZBB),Y ;set Control & Command registers
F415 999302 3105 STA X0293,Y ;and possible non-standard baud rate
F418 C8 3106 INY
F419 C004 3107 CPY $04
F41B D0F2 3108 BNE BF40F
F41D 204AEF 3109 BF41D JSR SEF4A ;fetch word length and move to
F420 8E9802 3110 STX X0298 ;number of bits to send
F423 AD9302 3111 LDA X0293 ;check RS-232 Control Register
F426 290F 3112 AND $0F
F428 F01C 3113 BEQ BF446 ;if standard baud rate specified,
F42A 0A 3114 ASL A ;calculate index into table
F42B AA 3115 TAX
F42C ADA602 3116 LDA X02A6 ;if US machine
F42F D009 3117 BNE BF43A
F431 BCC1FE 3118 LDY TFEC2-1,X ;use US table of baud rate
F434 BDC0FE 3119 LDA TFEC2-2,X
F437 4C40F4 3120 JMP JF440 ;go set baud rate
3121 ;
F43A BCEBE4 3122 BF43A LDY TE4EC-1,X ;use Intl table of baud rate factors
F43D BDEAE4 3123 LDA TE4EC-2,X ;to fetch baud rate factor
F440 8C9602 3124 JF440 STY X0296 ;set baud rate factor
F443 8D9502 3125 STA X0295 ;low byte also
F446 AD9502 3126 BF446 LDA X0295
F449 0A 3127 ASL A
F44A 202EFF 3128 JSR SFF2E ;continue to compute baud rate
F44D AD9402 3129 LDA X0294 ;check RS-232 Command Register
F450 4A 3130 LSR A ;for 3-line or X-line mode
F451 9009 3131 BCC BF45C ;and skip if in 3-line mode
F453 AD01DD 3132 LDA XDD01
F456 0A 3133 ASL A
F457 B003 3134 BCS BF45C ;continue if DSR present, else
F459 200DF0 3135 JSR BF00D ;DSR Signal Missing Error
3136 ;
F45C AD9B02 3137 BF45C LDA X029B ;move RS-232 receive buffer input ptr
F45F 8D9C02 3138 STA X029C ;to RS-232 receive buffer output ptr
F462 AD9E02 3139 LDA X029E ;move RS-232 transmit buffer input ptr
F465 8D9D02 3140 STA X029D ;to RS-232 transmit buffer output ptr
F468 2027FE 3141 JSR SFE27 ;fetch top of memory in XY
F46B A5F8 3142 LDA ZF8 ;if high byte of RS-232 receive buffer
F46D D005 3143 BNE BF474 ;base address not initialized yet,
F46F 88 3144 DEY ;subtract 1 from top of memory address
F470 84FB 3145 STY ZF8 ;use top of memory address to
F472 86F7 3146 STX ZF7 ;initialize receive buffer base address
F474 A5FA 3147 BF474 LDA ZFA ;same operation for transmit buffer
F476 D005 3148 BNE JF47D ;base address
F478 88 3149 DEY
F479 84FA 3150 STY ZFA
F47B 86F9 3151 STX ZF9
F47D 38 3152 JF47D SEC
F47E A9F0 3153 LDA $F0
F480 4C2DFE 3154 JMP JFE2D ;reset top of memory pointers from XY

```

```

3156 ;initialize CIA2
3157 ;
F483 A97F 3158 SF483 LDA $7F
F485 8D0DD 3159 STA XDD0D ;clear any pending IRQ's in ICR2
F488 A906 3160 LDA $06
F48A 8D03DD 3161 STA XDD03 ;PB2 bits 1 and 2 as outputs
F48D 8D01DD 3162 STA XDD01 ;set DTR and RTS high
F490 A904 3163 LDA $04
F492 0D00DD 3164 ORA XDD00
F495 8D00DD 3165 STA XDD00 ;set RS-232 output to high (idle)
F498 A000 3166 LDY $00
F49A 8CA102 3167 STY X02A1 ;clear ICR2 activity register
F49D 60 3168 RTS
3169 ;
3170 ;load RAM from a device
3171 ;
F49E 86C3 3172 JF49E STX ZC3 ;set destination address from XY
F4A0 84C4 3173 STY ZC4
F4A2 6C3003 3174 JMP (X0330) ;load RAM (normally F4A5)
3175 ;
3176 ;standard load RAM entry
3177 ;
F4A5 8593 3178 WF4A5 STA Z93 ;set load/verify switch to load
F4A7 A900 3179 LDA $00
F4A9 8590 3180 STA Z90 ;clear ST
F4AB A5BA 3181 LDA ZBA ;if current device is the keyboard (0)
F4AD D003 3182 BNE BF4B2
F4AF 4C13F7 3183 BF4AF JMP JF713 ;indicate Illegal Device # Error
3184 ;
F4B2 C903 3185 BF4B2 CMP $03 ;if current device is the screen
F4B4 F0F9 3186 BEQ BF4AF ;indicate error
F4B6 907B 3187 BCC BF533 ;if not serial bus device
F4B8 A4B7 3188 LDY ZB7 ;and if no file name,
F4BA D003 3189 BNE BF4BF
F4BC 4C10F7 3190 JMP JF710 ;indicate File Name Missing Error
3191 ;
F4BF A6B9 3192 BF4BF LDX ZB9 ;move X to secondary address
F4C1 20AFF5 3193 JSR SF5AF ;handle load messages
F4C4 A960 3194 LDA $60 ;set current secondary address
F4C6 85B9 3195 STA ZB9
F4C8 20D5F3 3196 JSR SF3D5 ;perform open of serial bus device
F4CB A5BA 3197 LDA ZBA ;let A = current device
F4CD 2009ED 3198 JSR SED09 ;send TALK on serial bus
F4D0 A5B9 3199 LDA ZB9 ;fetch secondary address
F4D2 20C7ED 3200 JSR SEDC7 ;and send on serial bus
F4D5 2013EE 3201 JSR JEE13 ;input a byte on serial bus
F4D8 85AE 3202 STA ZAE ;set I/O end address
F4DA A590 3203 LDA Z90
F4DC 4A 3204 LSR A
F4DD 4A 3205 LSR A
F4DE B050 3206 BCS BF530 ;if ST doesn't indicate a timeout (read)
F4E0 2013EE 3207 JSR JEE13 ;input a byte on serial bus
F4E3 85AF 3208 STA ZAF ;set high byte of end address
F4E5 8A 3209 TXA
F4E6 D008 3210 BNE BF4F0 ;if EOL is not low,
F4E8 A5C3 3211 LDA ZC3 ;use destination address
F4EA 85AE 3212 STA ZAE ;as end address
F4EC A5C4 3213 LDA ZC4 ;ditto for high byte
F4EE 85AF 3214 STA ZAF
F4F0 20D2F5 3215 BF4F0 JSR SF5D2 ;print LOAD or VERIFY

```

F4F3	A9FD	3216	BF4F3	LDA \$FD	;clear timeout (read) bit
F4F5	2590	3217		AND Z90	;in ST
F4F7	8590	3218		STA Z90	
F4F9	20E1FF	3219		JSR SFFE1	;check for Stop key
F4FC	D003	3220		BNE BF501	;if depressed
F4FE	4C33F6	3221		JMP JF633	;abort load
		3222			
F501	2013EE	3223	BF501	JSR JEE13	;input a byte on serial bus
F504	AA	3224		TAX	
F505	A590	3225		LDA Z90	;if Timeout (read) set in ST
F507	4A	3226		LSR A	
F508	4A	3227		LSR A	
F509	B0E8	3228		BCS BF4F3	;abort load
F50B	8A	3229		TXA	
F50C	A493	3230		LDY Z93	;if in verify mode
F50E	F00C	3231		BEQ BF51C	
F510	A000	3232		LDY \$00	
F512	D1AE	3233		CMP (ZAE),Y	;compare byte read to memory
F514	F008	3234		BEQ BF51E	
F516	A910	3235		LDA \$10	
F518	201CFE	3236		JSR SFE1C	;and set verify error upon mismatch
F51B	2C	3237		.BY \$2C	;skip next instruction
F51C	91AE	3238	BF51C	STA (ZAE),Y	;load byte to memory
F51E	E6AE	3239	BF51E	INC ZAE	;bump load address
F520	D002	3240		BNE BF524	
F522	E6AF	3241		INC ZAF	
F524	2490	3242	BF524	BIT Z90	;if not end of file,
F526	50CB	3243		BVC BF4F3	;repeat
F528	20EFED	3244		JSR SEDEF	;else sent TALK on serial bus
F52B	2042F6	3245		JSR SF642	;close serial bus
F52E	9079	3246		BCC BF5A9	;and exit
F530	4C04F7	3247	BF530	JMP JF704	;indicate File Not Found Error
		3248			
F533	4A	3249	BF533	LSR A	;if input device is not 1 (cassette)
F534	B003	3250		BCS BF539	
F536	4C13F7	3251		JMP JF713	;indicate Illegal Device #
		3252			
F539	20D0F7	3253	BF539	JSR SF7D0	;fetch tape buffer pointer
F53C	B003	3254		BCS BF541	
F53E	4C13F7	3255		JMP JF713	;if invalid, indicate Illegal Device #
		3256			
F541	2017F8	3257	BF541	JSR SF817	;display msgs and test buttons for read
F544	B068	3258		BCS BF5AE	
F546	20AFF5	3259		JSR SF5AF	;handle load messages
F549	A5B7	3260	BF549	LDA ZB7	;if file name present
F54B	F009	3261		BEQ BF556	
F54D	20EAF7	3262		JSR SF7EA	;search tape for file name
F550	900B	3263		BCC BF55D	;if no errors, continue
F552	F05A	3264		BEQ BF5AE	;exit if end of tape
F554	B0DA	3265		BCS BF530	;error if not found
F556	202CF7	3266	BF556	JSR SF72C	;since no file name, get next tape hdr
F559	F053	3267		BEQ BF5AE	;exit if end of tape found
F55B	B0D3	3268		BCS BF530	;indicate File Not Found Error
F55D	A590	3269	BF55D	LDA Z90	;check ST for unrecoverable read error
F55F	2910	3270		AND \$10	
F561	38	3271		SEC	
F562	D04A	3272		BNE BF5AE	;and exit if so
F564	E001	3273		CPX \$01	;if not Program Header
F566	F011	3274		BEQ BF579	
F568	E003	3275		CPX \$03	

F56A D0DD	3276	BNE BF549	
F56C A001	3277	BF56C LDY \$01	
F56E B1B2	3278	LDA (ZB2),Y	
F570 85C3	3279	STA ZC3	;reset load address from tape buffer
F572 C8	3280	INY	
F573 B1B2	3281	LDA (ZB2),Y	;high byte also
F575 85C4	3282	STA ZC4	
F577 B004	3283	BCS BF57D	
F579 A5B9	3284	BF579 LDA ZB9	
F57B D0EF	3285	BNE BF56C	
F57D A003	3286	BF57D LDY \$03	;index low byte of end address
F57F B1B2	3287	LDA (ZB2),Y	
F581 A001	3288	LDY \$01	
F583 F1B2	3289	SBC (ZB2),Y	;compute length of block to load
F585 AA	3290	TAX	
F586 A004	3291	LDY \$04	
F588 B1B2	3292	LDA (ZB2),Y	
F58A A002	3293	LDY \$02	
F58C F1B2	3294	SBC (ZB2),Y	
F58E A8	3295	TAY	
F58F 18	3296	CLC	
F590 8A	3297	TXA	
F591 65C3	3298	ADC ZC3	
F593 85AE	3299	STA ZAE	;and set end address of I/O area
F595 98	3300	TYA	
F596 65C4	3301	ADC ZC4	
F598 85AF	3302	STA ZAF	
F59A A5C3	3303	LDA ZC3	
F59C 85C1	3304	STA ZC1	;set tape load address
F59E A5C4	3305	LDA ZC4	
F5A0 85C2	3306	STA ZC2	
F5A2 20D2F5	3307	JSR SF5D2	;display load messages
F5A5 204AF8	3308	JSR SF84A	;load from cassette
F5A8 24	3309	.BY \$24	;skip next instruction
F5A9 18	3310	BF5A9 CLC	;clear error flag
F5AA A6AE	3311	LDX ZAE	;exit with end address in XY
F5AC A4AF	3312	LDY ZAF	
F5AE 60	3313	BF5AE RTS	


```

3315 ;handle messages for a load operation
3316 ;
F5AF A59D 3317 SF5AF LDA Z9D
F5B1 101E 3318 BPL BF5D1 ;exit if in Run mode
F5B3 A00C 3319 LDY $0C
F5B5 202FF1 3320 JSR BF12F ;print SEARCHING
F5B8 A5B7 3321 LDA ZB7 ;if file name is present,
F5BA F015 3322 BEQ BF5D1
F5BC A017 3323 LDY $17
F5BE 202FF1 3324 JSR BF12F ;print FOR
F5C1 A4B7 3325 JF5C1 LDY ZB7 ;if file name is present
F5C3 F00C 3326 BEQ BF5D1
F5C5 A000 3327 LDY $00
F5C7 B1BB 3328 BF5C7 LDA (ZBB),Y ;print file name
F5C9 20D2FF 3329 JSR SFFD2
F5CC C8 3330 INY
F5CD C4B7 3331 CPY ZB7
F5CF D0F6 3332 BNE BF5C7 ;repeat for all characters in name
F5D1 60 3333 BF5D1 RTS
3334 ;
3335 ;handle Load/Verify message
3336 ;
F5D2 A049 3337 SF5D2 LDY $49 ;assume Load
F5D4 A593 3338 LDA Z93
F5D6 F002 3339 BEQ BF5DA
F5D8 A059 3340 LDY $59 ;change to Verify if flag is non-zero
F5DA 4C2BF1 3341 BF5DA JMP JF12B ;display message and return

```

```

3343 ; Move RAM to a device
3344 ;
F5DD 86AE 3345 JF5DD STX ZAE ;set end of I/O area from XY
F5DF 84AF 3346 STY ZAF
F5E1 AA 3347 TAX ;A = index to start address
F5E2 B500 3348 LDA Z00,X
F5E4 85C1 3349 STA ZC1 ;move start address to ZC1/2
F5E6 B501 3350 LDA Z01,X
F5E8 85C2 3351 STA ZC2
F5EA 6C3203 3352 JMP (X0332) ;perform Save (normally F5ED)
3353 ;
3354 ;standard save RAM entry
3355 ;
F5ED A5BA 3356 WF5ED LDA ZBA ;if current device is keyboard (0)
F5EF D003 3357 BNE BF5F4
F5F1 4C13F7 3358 BF5F1 JMP JF713 ;indicate Illegal Device #
3359 ;
F5F4 C903 3360 BF5F4 CMP $03 ;if current device is screen
F5F6 F0F9 3361 BEQ BF5F1 ;indicate error
F5F8 905F 3362 BCC BF659 ;skip if cassette
F5FA A961 3363 LDA $61 ;device must be serial bus,
F5FC 85B9 3364 STA ZB9 ;set temporary sec. adr to Output
F5FE A4B7 3365 LDY ZB7 ;if file name is not present
F600 D003 3366 BNE BF605
F602 4C10F7 3367 JMP JF710 ;indicate File Name Missing
3368 ;
F605 20D5F3 3369 BF605 JSR SF3D5 ;perform open for serial bus device
F608 208FF6 3370 JSR SF68F ;print Saving + file name
F60B A5BA 3371 LDA ZBA ;get current device
F60D 200CED 3372 JSR SED0C ;and send LISTEN on serial bus
F610 A5B9 3373 LDA ZB9 ;send secondary address
F612 20B9ED 3374 JSR SEDB9 ;on serial bus
F615 A000 3375 LDY $00
F617 208EFB 3376 JSR SFB8E ;move ptr to I/O area into ZAC/D
F61A A5AC 3377 LDA ZAC
F61C 20DDED 3378 JSR JEDDD ;send low of start address on serial bus
F61F A5AD 3379 LDA ZAD
F621 20DDED 3380 JSR JEDDD ;then high byte
F624 20D1FC 3381 BF624 JSR SFCDD ;if block is not finished
F627 B016 3382 BCS BF63F
F629 B1AC 3383 LDA (ZAC),Y ;fetch next byte
F62B 20DDED 3384 JSR JEDDD ;and send on serial bus
F62E 20E1FF 3385 JSR SFEE1 ;check for Stop key
F631 D007 3386 BNE BF63A
F633 2042F6 3387 JF633 JSR SF642 ;and close serial bus if depressed
F636 A900 3388 LDA $00
F638 38 3389 SEC
F639 60 3390 RTS
3391 ;
F63A 20DBFC 3392 BF63A JSR SFCDB ;increment address
F63D D0E5 3393 BNE BF624 ;and repeat
F63F 20FEED 3394 BF63F JSR SEDFE ;send UNLISTEN on the serial bus

```

```

3396 ;close serial bus device
3397 ;
F642 24B9 3398 SF642 BIT ZB9 ;if secondary address valid,
F644 3011 3399 BMI BF657
F646 A5BA 3400 LDA ZBA ;get current device
F648 200CED 3401 JSR SEDOC ;and send LISTEN on serial bus
F64B A5B9 3402 LDA ZB9 ;fetch secondary address
F64D 29EF 3403 AND $EF ;strip bit 4
F64F 09E0 3404 ORA $E0 ;force bits 7-5
F651 20B9ED 3405 JSR SEDB9 ;send secondary addr after LISTEN
F654 20FEED 3406 JF654 JSR SEDFE ;send UNLISTEN on serial bus
F657 18 3407 BF657 CLC
F658 60 3408 RTS
3409 ;
F659 4A 3410 BF659 LSR A ;if device is not cassette
F65A B003 3411 BCS BF65F
F65C 4C13F7 3412 JMP JF713 ;indicate Illegal Device #
3413 ;
3414 ;save RAM to cassette
3415 ;
F65F 20D0F7 3416 BF65F JSR SF7D0 ;set tape buffer pointer in XY
F662 908D 3417 BCC BF5F1 ;exit if address invalid
F664 2038F8 3418 JSR SF838 ;print msgs for cassette and test sense
F667 B025 3419 BCS BF68E
F669 208FF6 3420 JSR SF68F ;print message Saving + file name
F66C A203 3421 LDX $03
F66E A5B9 3422 LDA ZB9
F670 2901 3423 AND $01 ;if secondary address is even,
F672 D002 3424 BNE BF676
F674 A201 3425 LDX $01 ;set secondary address to 1
F676 8A 3426 BF676 TXA
F677 206AF7 3427 JSR SF76A ;write Program Header block to cassette
F67A B012 3428 BCS BF68E ;exit upon error
F67C 2067F8 3429 JSR SF867 ;write block to cassette
F67F B00D 3430 BCS BF68E ;again, exit upon error
F681 A5B9 3431 LDA ZB9 ;if write End-Of-Tape Mark specified,
F683 2902 3432 AND $02
F685 F006 3433 BEQ BF68D
F687 A905 3434 LDA $05 ;then write special block
F689 206AF7 3435 JSR SF76A
F68C 24 3436 .BY $24 ;skip next instruction
F68D 18 3437 BF68D CLC ;indicate no errors
F68E 60 3438 BF68E RTS
3439 ;
3440 ;handle message Saving plus file name
3441 ;
F68F A59D 3442 SF68F LDA Z9D
F691 10FB 3443 BPL BF68E ;return if not in Direct mode
F693 A051 3444 LDY $51 ;point to message SAVING
F695 202FF1 3445 JSR BF12F ;print message
F698 4CC1F5 3446 JMP JF5C1 ;then print file name

```

```

3448 ;increment real time clock
3449 ;
F69B A200 3450 JF69B LDX $00
F69D E6A2 3451 INC ZA2 ;bump low byte
F69F D006 3452 BNE BF6A7
F6A1 E6A1 3453 INC ZA1 ;then middle byte upon overflow
F6A3 D002 3454 BNE BF6A7
F6A5 E6A0 3455 INC ZA0 ;then high byte upon overflow
F6A7 38 3456 BF6A7 SEC
F6A8 A5A2 3457 LDA ZA2 ;check for full 24 hours
F6AA E901 3458 SBC $01
F6AC A5A1 3459 LDA ZA1
F6AE E91A 3460 SBC $1A
F6B0 A5A0 3461 LDA ZA0
F6B2 E94F 3462 SBC $4F
F6B4 9006 3463 BCC BF6BC
F6B6 86A0 3464 STX ZA0 ;yes, clear clock
F6B8 86A1 3465 STX ZA1
F6BA 86A2 3466 STX ZA2
F6BC AD01DC 3467 BF6BC LDA XDC01 ;read current keyboard scan line
F6BF CD01DC 3468 CMP XDC01
F6C2 D0F8 3469 BNE BF6BC ;wait until steady
F6C4 AA 3470 TAX
F6C5 3013 3471 BMI BF6DA
F6C7 A2BD 3472 LDX $BD ;select Stop key row
F6C9 8E00DC 3473 STX XDC00
F6CC AE01DC 3474 BF6CC LDX XDC01 ;read column
F6CF EC01DC 3475 CPX XDC01
F6D2 D0F8 3476 BNE BF6CC ;wait until steady
F6D4 8D00DC 3477 STA XDC00
F6D7 E8 3478 INX
F6D8 D002 3479 BNE BF6DC
F6DA 8591 3480 BF6DA STA Z91 ;save scan result
F6DC 60 3481 BF6DC RTS
3482 ;
3483 ;read real time clock
3484 ;
F6DD 78 3485 JF6DD SEI ;kill IRQ to avoid interference
F6DE A5A2 3486 LDA ZA2 ;read low byte into A
F6E0 A6A1 3487 LDX ZA1 ;read middle byte into X
F6E2 A4A0 3488 LDY ZA0 ;and high byte into Y
3489 ;
3490 ;set real time clock
3491 ;
F6E4 60 3492 JF6E4 SEI ;disable IRQ to avoid interference
F6E5 85A2 3493 STA ZA2 ;set low byte from A
F6E7 86A1 3494 STX ZA1 ;set middle byte from X
F6E9 84A0 3495 STY ZA0 ;and set high byte from Y
F6EB 58 3496 CLI ;enable IRQ's
F6EC 60 3497 RTS

```

```

3499 ;test STOP key
3500 ;
F6ED A591 3501 WF6ED LDA Z91
F6EF C97F 3502 CMP $7F ;if Stop key is depressed
F6F1 D007 3503 BNE BF6FA
F6F3 08 3504 PHP ;save flags on stack
F6F4 20CCFF 3505 JSR SFFCC ;close files and set devices to default
F6F7 85C6 3506 STA ZC6 ;cancel keyboard queue
F6F9 28 3507 PLP ;restore flags
F6FA 60 3508 BF6FA RTS
3509 ;
3510 ;handle I/O errors
3511 ;
F6FB A901 3512 JF6FB LDA $01 ;Too Many Files
F6FD 2C 3513 .BY $2C
F6FE A902 3514 JF6FE LDA $02 ;File Open
F700 2C 3515 .BY $2C
F701 A903 3516 JF701 LDA $03 ;File Not Open
F703 2C 3517 .BY $2C
F704 A904 3518 JF704 LDA $04 ;File Not Found
F706 2C 3519 .BY $2C
F707 A905 3520 JF707 LDA $05 ;Device Not Present
F709 2C 3521 .BY $2C
F70A A906 3522 JF70A LDA $06 ;Not Input File
F70C 2C 3523 .BY $2C
F70D A907 3524 JF70D LDA $07 ;Not Output File
F70F 2C 3525 .BY $2C
F710 A908 3526 JF710 LDA $08 ;File Name Missing
F712 2C 3527 .BY $2C
F713 A909 3528 JF713 LDA $09 ;Illegal Device #
F715 48 3529 PHA ;save error # on stack
F716 20CCFF 3530 JSR SFFCC ;close files and set devices to default
F719 A000 3531 LDY $00
F71B 249D 3532 BIT Z9D
F71D 500A 3533 BVC BF729 ;if bit 6 in Direct/Run flag is set
F71F 202FF1 3534 JSR BF12F ;print message I/O Error
F722 68 3535 PLA
F723 48 3536 PHA
F724 0930 3537 ORA $30
F726 20D2FF 3538 JSR SFFD2 ;followed by error #
F729 68 3539 BF729 PLA ;restore error number
F72A 38 3540 SEC ;set error flag
F72B 60 3541 RTS

```

```

3543 ;get next file header from cassette
3544 ;
F72C A593 3545 SF72C LDA Z93 ;save load/verify switch on stack
F72E 48 3546 PHA
F72F 2041F8 3547 JSR SF841 ;read a block from tape
F732 68 3548 PLA
F733 8593 3549 STA Z93 ;restore load/verify flag
F735 B032 3550 BCS BF769 ;exit if read error
F737 A000 3551 LDY $00
F739 B1B2 3552 LDA (ZB2),Y ;get first character in tape buffer
F73B C905 3553 CMP $05 ;if code for End of Tape
F73D F02A 3554 BEQ BF769 ;return
F73F C901 3555 CMP $01
F741 F008 3556 BEQ BF74B ;if not code for Program Header
F743 C903 3557 CMP $03 ;or "?"
F745 F004 3558 BEQ BF74B
F747 C904 3559 CMP $04
F749 D0E1 3560 BNE SF72C ;or Data Header, try next block
F74B AA 3561 BF74B TAX
F74C 249D 3562 BIT Z9D ;if in direct mode,
F74E 1017 3563 BPL BF767
F750 A063 3564 LDY $63 ;point to message FOUND
F752 202FF1 3565 JSR BF12F ;and print it
F755 A005 3566 LDY $05
F757 B1B2 3567 BF757 LDA (ZB2),Y
F759 20D2FF 3568 JSR SFFD2 ;print a file name character
F75C C8 3569 INY
F75D C015 3570 CPY $15 ;and repeat for
F75F D0F6 3571 BNE BF757 ;all 21 characters
F761 A6:2 4683 FC872 DU: B:2
C874 35T5T0 4684 VES ETOT5 Xltfubiz ptba eiija prn jid. 8.5 seconds
F766 EA 3574 NOP ;filler for patch
F767 18 3575 BF767 CLC
F768 88 3576 DEY
F769 60 3577 BF769 RTS

```

```

3579 ;write a special block to cassette with code from A
3580 ;
F76A 859E 3581 SF76A STA Z9E ;save code from A
F76C 20D0F7 3582 JSR SF7D0 ;set tape buffer index in XY
F76F 905E 3583 BCC BF7CF ;exit if address invalid
F771 A5C2 3584 LDA ZC2 ;save I/O start address on stack
F773 48 3585 PHA
F774 A5C1 3586 LDA ZC1
F776 48 3587 PHA
F777 A5AF 3588 LDA ZAF ;save ptr to end of I/O area on stack
F779 48 3589 PHA
F77A A5AE 3590 LDA ZAE
F77C 48 3591 PHA
F77D A0BF 3592 LDY $BF
F77F A920 3593 LDA $20
F781 91B2 3594 BF781 STA (ZB2),Y ;clear cassette buffer with spaces
F783 88 3595 DEY
F784 D0FB 3596 BNE BF781
F786 A59E 3597 LDA Z9E ;move routine entry code
F788 91B2 3598 STA (ZB2),Y ;into first byte of cassette buffer
F78A C8 3599 INY
F78B A5C1 3600 LDA ZC1 ;move pointer to I/O area
F78D 91B2 3601 STA (ZB2),Y ;+ 1
F78F C8 3602 INY
F790 A5C2 3603 LDA ZC2
F792 91B2 3604 STA (ZB2),Y ;and + 2
F794 C8 3605 INY
F795 A5AE 3606 LDA ZAE ;move pointer to end of I/O area into
F797 91B2 3607 STA (ZB2),Y ;+ 3
F799 C8 3608 INY
F79A A5AF 3609 LDA ZAF
F79C 91B2 3610 STA (ZB2),Y ;and + 4
F79E C8 3611 INY
F79F 849F 3612 STY Z9F ;save index into tape buffer
F7A1 A000 3613 LDY $00
F7A3 849E 3614 STY Z9E ;initialize file name index
F7A5 A49E 3615 BF7A5 LDY Z9E
F7A7 C4B7 3616 CPY ZB7 ;if not at end of file name
F7A9 F00C 3617 BEQ BF7B7
F7AB B1BB 3618 LDA (ZBB),Y ;move file name
F7AD A49F 3619 LDY Z9F
F7AF 91B2 3620 STA (ZB2),Y ;into tape buffer
F7B1 E69E 3621 INC Z9E ;increment indexes
F7B3 E69F 3622 INC Z9F
F7B5 D0EE 3623 BNE BF7A5 ;and repeat
F7B7 20D7F7 3624 BF7B7 JSR SF7D7 ;set cassette buffer to I/O area
F7BA A969 3625 LDA $69
F7BC 85AB 3626 STA ZAB ;set 105 sync patterns
F7BE 206BF8 3627 JSR SF86B ;write block to tape
F7C1 A8 3628 TAY
F7C2 68 3629 PLA
F7C3 85AE 3630 STA ZAE ;restore ptr to beginning of I/O area
F7C5 68 3631 PLA
F7C6 85AF 3632 STA ZAF
F7C8 68 3633 PLA
F7C9 85C1 3634 STA ZC1 ;restore pointer to end of I/O area
F7CB 68 3635 PLA
F7CC 85C2 3636 STA ZC2
F7CE 98 3637 TYA
F7CF 60 3638 BF7CF RTS

```

```

3640 ;set tape buffer pointer in XY
3641 ;
F7D0 A6B2 3642 SF7D0 LDX ZB2 ;move X to low part of tape buffer ptr
F7D2 A4B3 3643 LDY ZB3 ;and Y to high part
F7D4 C002 3644 CPY $02 ;set C if address invalid
F7D6 60 3645 RTS
3646 ;
3647 ;set cassette buffer to I/O area
3648 ;
F7D7 20D0F7 3649 SF7D7 JSR SF7D0 ;set tape buffer address in XY
F7DA 8A 3650 TXA
F7DB 85C1 3651 STA ZC1 ;move to I/O start address
F7DD 18 3652 CLC
F7DE 69C0 3653 ADC $C0 ;add 192
F7E0 85AE 3654 STA ZAE ;to form low part of end of I/O area
F7E2 98 3655 TYA
F7E3 85C2 3656 STA ZC2 ;move high byte also
F7E5 6900 3657 ADC $00
F7E7 85AF 3658 STA ZAF
F7E9 60 3659 RTS
3660 ;
3661 ;search tape for a file name
3662 ;
F7EA 202CF7 3663 SF7EA JSR SF72C ;read next file header from tape
F7ED B01D 3664 BCS BF80C ;exit upon error
F7EF A005 3665 LDY $05
F7F1 849F 3666 STY Z9F ;set temporary index in tape buffer
F7F3 A000 3667 LDY $00
F7F5 849E 3668 STY Z9E ;set temporary index in file name
F7F7 C4B7 3669 BF7F7 CPY ZB7 ;if at end of file name
F7F9 F010 3670 BEQ BF80B ;then name found
F7FB B1BB 3671 LDA (ZBB),Y ;get character from file name
F7FD A49F 3672 LDY Z9F
F7FF D1B2 3673 CMP (ZB2),Y ;and compare to tape buffer
F801 D0E7 3674 BNE SF7EA ;if not equal, try next entry from tape
F803 E69E 3675 INC Z9E ;else bump pointers
F805 E69F 3676 INC Z9F
F807 A49E 3677 LDY Z9E
F809 D0EC 3678 BNE BF7F7 ;and repeat test
F80B 18 3679 BF80B CLC ;clear error flag
F80C 60 3680 BF80C RTS
3681 ;
3682 ;add 1 to tape index and test for overflow
3683 ;
F80D 20D0F7 3684 SF80D JSR SF7D0 ;get tape buffer pointer in XY
F810 E6A6 3685 INC ZA6 ;and add 1
F812 A4A6 3686 LDY ZA6
F814 C0C0 3687 CPY $C0 ;set C if end of block
F816 60 3688 RTS

```



```

3690 ;handle messages and rest cassette buttons for read
3691 ;
F817 202EF8 3692 SF817 JSR SF82E ;test sense line
F81A F01A 3693 BEQ BF836 ;if no buttons depressed,
F81C A01B 3694 LDY $1B ;point to message Press Play on Tape
F81E 202FF1 3695 BF81E JSR BF12F ;and print message
F821 20D0F8 3696 BF821 JSR SF8D0 ;test Stop key
F824 202EF8 3697 JSR SF82E ;test sense line
F827 D0F8 3698 BNE BF821 ;and repeat if no buttons depressed
F829 A06A 3699 LDY $6A ;point to message OK
F82B 4C2FF1 3700 JMP BF12F ;and print
3701 ;
3702 ;test sense line for a button depressed on cassette
3703 ;
F82E A910 3704 SF82E LDA $10 ;set mask for sense line
F830 2401 3705 BIT Z01 ;test 6510 I/O register
F832 D002 3706 BNE BF836 ;exit with Z clear if nothing depressed
F834 2401 3707 BIT Z01 ;else set Z
F836 18 3708 BF836 CLC
F837 60 3709 RTS
3710 ;
3711 ;set messages and test cassette sense line for output
3712 ;
F838 202EF8 3713 SF838 JSR SF82E ;test sense switches
F83B F0F9 3714 BEQ BF836 ;if no buttons depressed
F83D A02E 3715 LDY $2E ;index message Press Next & Play...
F83F D0DD 3716 BNE BF81E ;print message, test sense and return
3717 ;
3718 ;read a block from cassette
3719 ,
F841 A900 3720 SF841 LDA $00
F843 8590 3721 STA Z90 ;clear ST
F845 8593 3722 STA Z93 ;set load/verify switch to load
F847 20D7F7 3723 JSR SF7D7 ;set tape buffer to I/O area
F84A 2017F8 3724 SF84A JSR SF817 ;handle msgs and test sense for read
F84D B01F 3725 BCS BF86E
F84F 78 3726 SEI ;disable IRQ
F850 A900 3727 LDA $00
F852 85AA 3728 STA ZAA ;set gap
F854 85B4 3729 STA ZB4 ;set no sync established
F856 85B0 3730 STA ZB0 ;set no special speed correction yet
F858 859E 3731 STA Z9E ;initialize error log index for pass 1
F85A 859F 3732 STA Z9F ;and pass 2
F85C 859C 3733 STA Z9C ;set no byte available yet
F85E A990 3734 LDA $90 ;set Flag mask
F860 A20E 3735 LDX $0E ;index for cassette read IRQ address
F862 D011 3736 BNE BF875 ;JMP

```

```

3738 ;write a block to cassette
3739 ;
F864 20D7F7 3740 SF864 JSR SF7D7 ;initialize tape buffer pointer
F867 A914 3741 SF867 LDA $14
F869 85AB 3742 STA ZAB ;20 sync patterns
F86B 2038F8 3743 SF86B JSR SF838 ;test sense and display msgs for output
F86E B06C 3744 BF86E BCS BF8DC
F870 78 3745 SEI
F871 A982 3746 LDA $82 ;mask for ICR1 to honor TBl
F873 A208 3747 LDX $08 ;IRQ index for cassette write, part 1
3748 ;
3749 ;common code for cassette read & write
3750 ;
F875 A07F 3751 BF875 LDY $7F
F877 8C0DDC 3752 STY XDCOD ;clear any pending masks in ICR1
F87A 8D0DDC 3753 STA XDCOD ;then set mask for TBl
F87D ADOEDC 3754 LDA XDCOE
F880 0919 3755 ORA $19 ;+ force load, one shot and TBl to CRAl
F882 8D0FDC 3756 STA XDCOF ;to form CRBl
F885 2991 3757 AND $91
F887 8DA202 3758 STA X02A2 ;and CRBl activity register
F88A 20A4F0 3759 JSR SF0A4 ;condition flag bit in ICR2
F88D AD11D0 3760 LDA XD011
F890 29EF 3761 AND $EF
F892 8D11D0 3762 STA XD011 ;disable the screen
F895 AD1403 3763 LDA X0314 ;save standard IRQ vector
F898 8D9F02 3764 STA X029F
F89B AD1503 3765 LDA X0315
F89E 8DA002 3766 STA X02A0
F8A1 20BDFC 3767 JSR SFCBD ;set new IRQ for cassette depending on X
F8A4 A902 3768 LDA $02
F8A6 85BE 3769 STA ZBE ;select phase 2
F8A8 2097FB 3770 JSR SFB97 ;initialize cassette I/O variables
F8AB A501 3771 LDA Z01
F8AD 291F 3772 AND $1F
F8AF 8501 3773 STA Z01 ;start cassette motor
F8B1 85C0 3774 STA ZC0 ;set tape motor interlock
F8B3 A2FF 3775 LDX $FF
F8B5 A0FF 3776 BF8B5 LDY $FF
F8B7 88 3777 BF8B7 DEY
F8B8 D0FD 3778 BNE BF8B7 ;delay .3 seconds
F8BA CA 3779 DEX
F8BB D0F8 3780 BNE BF8B5
F8BD 58 3781 CLI
F8BE ADA002 3782 JF8BE LDA X02A0 ;test high byte of IRQ save area
F8C1 CD1503 3783 CMP X0315 ;to determine if end of I/O
F8C4 18 3784 CLC
F8C5 F015 3785 BEQ BF8DC ;exit if so
F8C7 20D0F8 3786 JSR SF8D0 ;else test Stop key
F8CA 20BCF6 3787 JSR BF6BC ;scan keyboard
F8CD 4CBEF8 3788 JMP JF8BE ;repeat

```

```

3790 ;handle Stop key during cassette operations
3791 ;
F8D0 20E1FF 3792 SF8D0 JSR SFF8E1 ;test Stop key
F8D3 18 3793 CLC
F8D4 D00E 3794 BIT BF8E1
F8D6 2093FC 3795 JSR SFC9E ;if depressed, stop cassette operations
F8D9 38 3796 SEC
F8DA 68 3797 PLA ;delete own return address
F8DB 68 3798 PLA
F8DC A900 3799 BF8DC LDA $00
F8DE 8DA002 3800 STA X02A0 ;clear high byte of 1k0 save area
F8E1 60 3801 BF8E1 RTS
3802 ;
3803 ;schedule CIA1 Timer A depending in parmater in X
3804 ;
F8E2 86B1 3805 SF8E2 STX ZB1 ;save entry parameter
F8E4 A5B0 3806 LDA ZB0 ;get speed correction
F8E6 0A 3807 ASL A ;* 2
F8E7 0A 3808 ASL A ;* 4
F8E8 18 3809 CLC
F8E9 65B0 3810 ADC ZB0 ;add speed correction
F8EB 18 3811 CLC
F8EC 65B1 3812 ADC ZB1 ;and parameter
F8EE 85B1 3813 STA ZB1 ;save low order
F8F0 A900 3814 LDA $00
F8F2 24B0 3815 BIT ZB0 ;if speed correction is positive
F8F4 3001 3816 BMI BF8F7
F8F6 2A 3817 ROL A ;set high order in A
F8F7 06B1 3818 BF8F7 ASL ZB1 -
F8F9 2A 3819 ROL A ;* 2
F8FA 06E1 3820 ASL ZB1
F8FC 2A 3821 ROL A ;* 4
F8FD AA 3822 TAX
F8FE AD06DC 3823 BF8FE LDA XDC06 ;wait until no chance of
F901 C916 3824 CMP $16 ;TBL1 changing
F903 90F9 3825 BCC BF8FE ;while it still must be read
F905 65E1 3826 ADC ZB1 ;add low order offset to TBL1
F907 8D04DC 3827 STA XDC04 ;and store in TAL1
F90A 8A 3828 TXA
F90B 6D07DC 3829 ADC XDC07 ;add high order offset to TBL1
F90E 8D05DC 3830 STA XDC05 ;and store in TAL1
F911 ADA202 3831 LDA X02A2
F914 8D0EDC 3832 STA XDC0E ;set CRL1 from CRB1 activity register
F917 8DA402 3833 STA X02A4 ;and save it
F91A AD0DDC 3834 LDA XDC0D
F91D 2910 3835 AND $10
F91F F009 3836 BEQ BF92A ;if Flag bit is not set,
F921 A9F9 3837 LDA >BF92A ;set exit address on stack
F923 48 3838 PHA
F924 A977 3839 BIT >BF92A
F926 48 3840 PHA
F927 4C43FF 3841 JMP JFF43 ;and simulate an IRQ
F92A 5E 3842 BF92A CLI ;else allow 1k0 and exit
F92B 60 3843 RTS

```

```

3845 ;cassette read IRQ routine
3846 ;
F92C 3847 WF92C = *
F92C AE07DC 3848 BF92C LDX XDC07 ;get TBH1
F92F A0FF 3849 LDY $FF
F931 98 3850 TYA ;and complement of TBH1
F932 ED06DC 3851 SBC XDC06 ;(time elapsed)
F935 EC07DC 3852 CPX XDC07 ;if high byte not steady,
F938 D0F2 3853 BNE BF92C ;repeat
F93A 86B1 3854 STX ZB1 ;else save high byte
F93C AA 3855 TAX
F93D 8C06DC 3856 STY XDC06 ;reset TBH1 to maximum
F940 8C07DC 3857 STY XDC07 ;ditto TBH1
F943 A919 3858 LDA $19 ;force load, one-shot and Timer B
F945 8D0FDC 3859 STA XDC0F ;into CRB1
F948 AD0DDC 3860 LDA XDC0D
F94B 8DA302 3861 STA X02A3 ;save ICRI
F94E 98 3862 TYA
F94F E5B1 3863 SBC ZB1 ;complement high byte
F951 86B1 3864 STX ZB1 ;save low byte
F953 4A 3865 LSR A ;elapsed time in A, ZB1
F954 66B1 3866 ROR ZB1 ;/ 2
F956 4A 3867 LSR A
F957 66B1 3868 ROR ZB1 ;/ 4
F959 A5B0 3869 LDA ZB0 ;get speed correction
F95B 18 3870 CLC
F95C 693C 3871 ADC $3C ;+ 240 microseconds
F95E C5B1 3872 CMP ZB1 ;if cycle shorter
F960 B04A 3873 BCS BF9AC ;dismiss
F962 A69C 3874 LDX Z9C ;if byte available
F964 F003 3875 BEQ BF969
F966 64C0FA 3876 JMP JFA60 ;receive it
3877 ;
F969 A6A3 3878 BF969 LDX ZA3 ;test bit count and if beyond last bit,
F96B 301B 3879 BMI BF988 ;do end of byte
F96D A200 3880 LDX $00 ;assume bit value of 0
F96F 6930 3881 ADC $30 ;add 432 microseconds
F971 65B0 3882 ADC ZB0 ;+ 2 * speed correction
F973 C5B1 3883 CMP ZB1 ;if cycle shorter
F975 B01C 3884 BCS BF993 ;record a 0
F977 E8 3885 INX ;assume bit value of 1
F978 6926 3886 ADC $26 ;get 584 microseconds
F97A 65B0 3887 ADC ZB0 ;+ 3 * speed correction
F97C C5B1 3888 CMP ZB1 ;if cycle shorter
F97E B017 3889 BCS BF997 ;record a 1
F980 692C 3890 ADC $2C ;get 760 microseconds
F982 65B0 3891 ADC ZB0 ;+ 4 * speed correction
F984 C5B1 3892 CMP ZB1 ;if cycle shorter
F986 9003 3893 BCC BF98B
F988 4C10FA 3894 BF988 JMP JFA10 ;go do end of byte
3895 ;
F98B A5B4 3896 BF98B LDA ZB4 ;if sync established
F98D F01D 3897 BEQ BF9AC
F98F 85A8 3898 STA ZA8 ;set erroneous bits
F991 D019 3899 BNE BF9AC
F993 E6A9 3900 BF993 INC ZA9 ;for a 0, increment 0/1 balance
F995 B002 3901 BCS BF999
F997 C6A9 3902 BF997 DEC ZA9 ;for a 1, decrement 0/1 balance
F999 38 3903 BF999 SEC
F99A E913 3904 SBC $13 ;0/1 cutoff level

```

F99C E5B1	3905	SBC ZB1	;- cycle width
F99E 6592	3906	ADC Z92	
F9A0 8592	3907	STA Z92	;accumulated for speed correction
F9A2 A5A4	3908	LDA ZA4	
F9A4 4901	3909	EOR \$01	;flip cycle indication
F9A6 85A4	3910	STA ZA4	
F9A8 F02B	3911	BEQ BF9D5	;if first cycle,
F9AA 86D7	3912	STX ZD7	;save bit value
F9AC A5B4	3913	BF9AC LDA ZB4	;if no sync yet
F9AE F022	3914	BEQ BF9D2	;return from IRQ
F9B0 ADA302	3915	LDA X02A3	;if ICRL mask
F9B3 2901	3916	AND \$01	
F9B5 D005	3917	BNE BF9BC	
F9B7 ADA402	3918	LDA X02A4	;and last CRL mask shows no TAL flag,
F9BA D016	3919	BNE BF9D2	;exit from IRQ
F9BC A900	3920	BF9BC LDA \$00	
F9BE 85A4	3921	STA ZA4	;clear cycle count
F9C0 8DA402	3922	STA X02A4	;and last CRL mask
F9C3 A5A3	3923	LDA ZA3	;if bit count indicates end of byte,
F9C5 1030	3924	BFL BF9F7	
F9C7 30BF	3925	BMI BF988	;go do end of byte
F9C9 A2A6	3926	BF9C9 LDX \$A6	
F9CB 20E2F8	3927	JSR SF8E2	;schedule timer
F9CE A59B	3928	LDA Z9B	;if parity calculated does not match
F9D0 D0B9	3929	BNE BF98B	;set erroneous bit flag
F9D2 4CBCFE	3930	BF9D2 JMP JFEBBC	;exit from IRQ
	3931	;	
F9D5 A592	3932	BF9D5 LDA Z92	;if second cycle,
F9D7 F007	3933	BEQ BF9E0	;check accumulated over/under time
F9D9 3003	3934	BMI BF9DE	
F9DB C6B0	3935	DEC ZB0	
F9DD 2C	3936	.BY \$2C	;skip next instruction
F9DE E6B0	3937	BF9DE INC ZB0	;adapt speed correction accordingly
F9E0 A900	3938	BF9E0 LDA \$00	
F9E2 8592	3939	STA Z92	;reset accumulated over/under time
F9E4 E4D7	3940	CPX ZD7	;if 2nd cycle = complement of cycle 1,
F9E6 D00F	3941	BNE BF9F7	;include bit
F9E8 8A	3942	TXA	
F9E9 D0A0	3943	BNE BF98B	;if two 0 cycles
F9EB A5A9	3944	LDA ZA9	;and 0/1 balance
F9ED 30BD	3945	BMI BF9AC	
F9EF C910	3946	CMP \$10	;at least 16 "0" cycles extra
F9F1 90B9	3947	BCC BF9AC	
F9F3 8596	3948	STA Z96	;set sync detected
F9F5 B0B5	3949	BCS BF9AC	
F9F7 8A	3950	BF9F7 TXA	
F9F8 459B	3951	EOR Z9B	;calculate parity
F9FA 859B	3952	STA Z9B	
F9FC A5B4	3953	LDA ZB4	;if no sync yet,
F9FE F0D2	3954	BEQ BF9D2	;exit
FA00 C6A3	3955	DEC ZA3	;decrement pending bit count
FA02 30C5	3956	BMI BF9C9	;after last bit, check parity
FA04 46D7	3957	LSR ZD7	;include bit
FA06 66BF	3958	ROR ZBF	;in byte being read
FA08 A2DA	3959	LDX \$DA	
FA0A 20E2F8	3960	JSR SF8E2	;schedule timer
FA0D 4CBCFE	3961	JMP JFEBBC	;exit from IRQ
	3962	;	
FA10 A596	3963	JFA10 LDA Z96	;if sync detected
FA12 F004	3964	BEQ BFA18	

FA14	A5B4	3965	LDA ZB4	;and not yet established
FA16	F007	3966	BEQ BFA1F	
FA18	A5A3	3967	BFA18 LDA ZA3	;or last bit done
FA1A	3003	3968	BMI BFA1F	
FA1C	4C97F9	3969	JMP BF997	;allow byte reception
		3970	;	
FA1F	46B1	3971	BFA1F LSR ZB1	;compute new speed correction value
FA21	A993	3972	LDA \$93	
FA23	38	3973	SEC	
FA24	E5B1	3974	SBC ZB1	
FA26	65B0	3975	ADC ZB0	
FA28	0A	3976	ASL A	
FA29	AA	3977	TAX	
FA2A	20E2F8	3978	JSR SF8E2	;schedule timer
FA2D	E69C	3979	INC Z9C	;indicate byte available
FA2F	A5B4	3980	LDA ZB4	;if no bit received in
FA31	D011	3981	BNE BFA44	
FA33	A596	3982	LDA Z96	;but sync detected
FA35	F026	3983	BEQ BFA5D	
FA37	85A8	3984	STA ZA8	;set error bits
FA39	A900	3985	LDA \$00	
FA3B	8596	3986	STA Z96	;clear sync detected
FA3D	A981	3987	LDA \$81	;set TA1 bit
FA3F	8D0DDC	3988	STA XDCOD	;in ICRI
FA42	85B4	3989	STA ZB4	;set sync established
FA44	A596	3990	BFA44 LDA Z96	;move sync status
FA46	85B5	3991	STA ZB5	;to saved sync status
FA48	F009	3992	BEQ BFA53	
FA4A	A900	3993	LDA \$00	;if not detected,
FA4C	85B4	3994	STA ZB4	;indicate sync not established
FA4E	A901	3995	LDA \$01	
FA50	8D0DDC	3996	STA XDCOD	;clear TA mask in ICRI
FA53	A596	3997	BFA53 LDA Z96	;save byte read
FA55	85BD	3998	STA ZBD	
FA57	A5A8	3999	LDA ZA8	
FA59	05A9	4000	ORA ZA9	;accumulate possible errors
FA5B	85B6	4001	STA ZB6	
FA5D	4CBCE	4002	BFA5D JMP JFEB	;exit from IRQ

```

4004 ;receive next byte from cassette
4005 ;
FA60 2097FB 4006 JFA60 JSR SFB97 ;initialize cassette I/O variables
FA63 859C 4007 STA Z9C ;indicate no byte available yet
FA65 A2DA 4008 LDX $DA
FA67 20E2F8 4009 JSR SF8E2 ;schedule Timer
FA6A A5BE 4010 LDA ZBE ;if first or second phase being read,
FA6C F002 4011 BEQ BFA70
FA6E 85A7 4012 STA ZA7 ;copy into actual phase
FA70 A90F 4013 BFA70 LDA $0F
FA72 24AA 4014 BIT ZAA ;if data beyond area
FA74 1017 4015 BPL BFA8D
FA76 A5B5 4016 LDA ZB5 ;and in second half
FA78 D00C 4017 BNE BFA86
FA7A A6BE 4018 LDX ZBE
FA7C CA 4019 DEX
FA7D D00B 4020 BNE BFA8A
FA7F A908 4021 LDA $08 ;indicate Long Block in ST
FA81 201CFE 4022 JSR SFE1C
FA84 D004 4023 BNE BFA8A ;and exit.
FA86 A900 4024 BFA86 LDA $00
FA88 85AA 4025 STA ZAA ;set gap
FA8A 4CBCFE 4026 BFA8A JMP JFEBC ;exit from IRQ
4027 ;
FA8D 7031 4028 BFA8D BVS BFAC0 ;if data, go collect
FA8F D018 4029 BNE BFAA9 ;if header, decrement count
FA91 A5B5 4030 LDA ZB5 ;if gap and byte valid
FA93 D0F5 4031 BNE BFA8A
FA95 A5B6 4032 LDA ZB6 ;and no erroneous bits
FA97 D0F1 4033 BNE BFA8A
FA99 A5A7 4034 LDA ZA7 ;get actual phase
FA9B 4A 4035 LSR A ;bit 7 of byte
FA9C A5BD 4036 LDA ZBD
FA9E 3003 4037 BMI BFAA3
FAA0 9018 4038 BCC BFABA
FAA2 18 4039 CLC
FAA3 B015 4040 BFAA3 BCS BFABA ;must correspond to header phase
FAA5 290F 4041 AND $0F
FAA7 85AA 4042 STA ZAA ;set header count
FAA9 C6AA 4043 BFAA9 DEC ZAA ;decrement header count
FAAB D0DD 4044 BNE BFA8A ;at end of header,
FAAD A940 4045 LDA $40
FAAF 85AA 4046 STA ZAA ;set data
FAB1 208EFB 4047 JSR SFB8E ;move beginning of I/O area into ZAC/D
FAB4 A900 4048 LDA $00
FAB6 85AB 4049 STA ZAB ;clear checksum
FAB8 F0D0 4050 BEQ BFABA
FABA A980 4051 BFABA LDA $80 ;if illegal header
FABC 85AA 4052 STA ZAA ;set Data After Area
FABE D0CA 4053 BNE BFA8A
FAC0 A5B5 4054 BFAC0 LDA ZB5 ;if data, but data not valid
FAC2 F00A 4055 BEQ BFACE
FAC4 A904 4056 LDA $04 ;set Short Block Error
FAC6 201CFE 4057 JSR SFE1C ;and indicate in ST
FAC9 A900 4058 LDA $00
FACB 4C4AFB 4059 JMP JFB4A ;do end of block
4060 ;
FACE 20D1FC 4061 BFACE JSR SFCD1 ;if at end of area,
FAD1 9003 4062 BCC BFAD6
FAD3 4C48FB 4063 JMP JFB48 ;set data after area

```

```

4064 ;
FAD6 A6A7 4065 BFAD6 LDX ZA7 ;if actual phase = 1
FAD8 CA 4066 DEX
FAD9 F02D 4067 BEQ BFB08 ;do error correction
FADB A593 4068 LDA Z93
FADD F00C 4069 BEQ BFAEB ;if Verify pass
FADF A000 4070 LDY $00
FAE1 A5BD 4071 LDA ZBD
FAE3 DIAC 4072 CMP (ZAC),Y ;compare byte to current character
FAE5 F004 4073 BEQ BFAEB
FAE7 A901 4074 LDA $01
FAE9 85B6 4075 STA ZB6 ;set error flag upon mismatch
FAEB A5B6 4076 BFAEB LDA ZB6 ;if errors
FAED F04B 4077 BEQ BFB3A
FAEF A23D 4078 LDX $3D
FAF1 E49E 4079 CPX Z9E ;and error log table is not at maximum
FAF3 903E 4080 BCC BFB33
FAF5 A69E 4081 LDX Z9E
FAF7 A5AD 4082 LDA ZAD
FAF9 9D0101 4083 STA X0100+1,X ;store pointer into error log table
FAFC A5AC 4084 LDA ZAC
FAFE 9D0001 4085 STA X0100,X
FB01 E8 4086 INX
FB02 E8 4087 INX
FB03 869E 4088 STX Z9E ;set new error log index
FB05 4C3AFB 4089 JMP BFB3A
4090 ;
FB08 A69F 4091 BFB08 LDX Z9F
FB0A E49E 4092 CPX Z9E ;if pass 2 error log index at maximum,
FB0C F03 4093 JFC FFFF ;no correction possible
FB0E A5AC 4094 LDA ZAC ;else
FB10 DD0001 4095 CMP X0100,X ;check for match in error log table
FB13 D02E 4096 BNE BFB43
FB15 A5AD 4097 LDA ZAD
FB17 DD0101 4098 CMP X0100+1,X
FB1A D027 4099 BNE BFB43
FB1C E69F 4100 INC Z9F ;if so, bump pass 2 error log index
FB1E E69F 4101 INC Z9F
FB20 A593 4102 LDA Z93 ;if Verify pass
FB22 F00B 4103 BEQ BFB2F
FB24 A5BD 4104 LDA ZBD
FB26 A000 4105 LDY $00
FB28 DIAC 4106 CMP (ZAC),Y ;compare byte to current character
FB2A F017 4107 BEQ BFB43 ;set error if not equal
FB2C C8 4108 INY
FB2D 84B6 4109 STY ZB6
FB2F A5B6 4110 BFB2F LDA ZB6 ;if still an error
FB31 F007 4111 BEQ BFB3A
FB33 A910 4112 BFB33 LDA $10
FB35 201CFE 4113 JSR SFE1C ;set Unrecoverable Read Error in ST
FB38 D009 4114 BNE BFB43
FB3A A593 4115 BFB3A LDA Z93 ;if Load,
FB3C D005 4116 BNE BFB43
FB3E A8 4117 TAY
FB3F A5BD 4118 LDA ZBD
FB41 91AC 4119 STA (ZAC),Y ;store byte
FB43 20DBFC 4120 BFB43 JSR SFCDB ;increment address
FB46 D043 4121 BNE BFB8B ;and exit
FB48 A980 4122 JFB48 LDA $80 ;set data after area
FB4A 85AA 4123 JFB4A STA ZAA

```


FB4C 78	4124	SEI	
FB4D A201	4125	LDX \$01	
FB4F 8E0DDC	4126	STX XDCOD	;clear TA mask from ICRI
FB52 AE0DDC	4127	LDX XDCOD	
FB55 A6BE	4128	LDX ZBE	;if phase 1 or 2
FB57 CA	4129	DEX	
FB58 3002	4130	BMI BFB5C	
FB5A 86BE	4131	STX ZBE	;decrement phase
FB5C C6A7	4132	BFB5C DEC ZA7	;decrement actual phase
FB5E F008	4133	BEQ BFB68	;if not end of block
FB60 A59E	4134	LDA Z9E	;and no errors encountered
FB62 D027	4135	BNE BFB8B	;skip phase
FB64 85BE	4136	STA ZBE	
FB66 F023	4137	BEQ BFB8B	;if end of block,
FB68 2093FC	4138	BFB68 JSR SFC93	;switch from cassette to default IRQ
FB6B 208EFB	4139	JSR SFB8E	;move beginning of I/O area into ZAC/D
FB6E A000	4140	LDY \$00	
FB70 84AB	4141	STY ZAB	;clear checksum value
FB72 B1AC	4142	BFB72 LDA (ZAC),Y	;compute checksum
FB74 45AB	4143	EOR ZAB	
FB76 85AB	4144	STA ZAB	
FB78 20DBFC	4145	JSR SFCDB	;bump address
FB7B 20D1FC	4146	JSR SFCD1	;check for end of block
FB7E 90F2	4147	BCC BFB72	
FB80 A5AB	4148	LDA ZAB	
FB82 45BD	4149	EOR ZBD	;if computed checksum does not match,
FB84 F005	4150	BEQ BFB8B	
FB86 A920	4151	LDA \$20	
FB88 201CFE	4152	JSR SFE1C	;set Checksum Error in ST
FB8B 4CBCFE	4153	BFB8B JMP JFEB C	;return from IRQ

```

4155 ;move save/load address into ZAC/D
4156 ;
FB8E A5C2 4157 SFB8E LDA ZC2      ;move high byte of save/load pointer
FB90 85AD 4158 STA ZAD
FB92 A5C1 4159 LDA ZC1      ;then low byte
FB94 85AC 4160 STA ZAC
FB96 60   4161 RTS
4162 ;
4163 ;initialize cassette read/write variables
4164 ;
FB97 A908 4165 SFB97 LDA $08      -
FB99 85A3 4166 STA ZA3      ;set bit count
FB9B A900 4167 LDA $00
FB9D 85A4 4168 STA ZA4      ;clear cycle count
FB9F 85A8 4169 STA ZA8      ;clear byte start cycle 1
FBA1 859B 4170 STA Z9B      ;clear parity bit
FBA3 85A9 4171 STA ZA9      ;clear byte start cycle 2
FBA5 60   4172 RTS
4173 ;
4174 ;schedule ICRL Timer B and invert cassette write line
4175 ;
FBA6 A5BD 4176 SFB66 LDA ZBD
FBA8 4A   4177 LSR A      ;move bit 0 of current character into C
FBA9 A960 4178 LDA $60      ;start with value for a 0
FBAE 9002 4179 BCC BFBAF
FBAD A9B0 4180 SFBAD LDA $B0    ;if a 1 to be sent, set value for a 1
FBAF A200 4181 BFBAF LDX $00
FBB1 8D06DC 4182 SFB11 STA XDC06 ;TB11 set to 0
FBB4 8E07DC 4183 STX XDC07 ;TBH1 set to $60/$B0
FBB7 AD0DDC 4184 LDA XDC0D
FBBA A919 4185 LDA $19      ;force load, one-shot and Timer B
FBBC 8D0FDC 4186 STA XDC0F ;into CRE1
FBBF A501 4187 LDA Z01
FBC1 4908 4188 EOR $08      ;invert polarity of cassette write bit
FBC3 8501 4189 STA Z01
FBC5 2908 4190 AND $08      ;indicate polarity in A and Z
FBC7 60   4191 RTS

```

```

4193 ;IRQ routine for cassette write, part 2 (at FBCE)
4194 ;
4195 ;used for byte start cycle and each half cycle per data bit
4196 ;
FBC8 38 4197 FBCE SEC
FBC9 66B6 4198 ROR ZB6 ;set current byte pointer => 32K
FBCB 303C 4199 BMI BFC09 ;JMP
FBCE A5A8 4200 WFBCE LDA ZA8 ;if byte start cycle 1
FBCF D012 4201 BNE BFBE3
FBD1 A910 4202 LDA $10
FBD3 A201 4203 LDX $01 ;set AX to 272
FBD5 20B1FB 4204 JSR SFBBI ;invert output and schedule timer
FBD8 D02F 4205 BNE BFC09 ;if output is zero again
FBDA E6A8 4206 INC ZA8 ;indicate cycle 1 finished
FBDC A5B6 4207 LDA ZB6 ;if current byte pointer => 32K
FBDE 1029 4208 BPL BFC09
FBE0 4C57FC 4209 JMP JFC57 ;switch to other cassette write IRQ
4210 ;
FBE3 A5A9 4211 BFBE3 LDA ZA9 ;if byte start cycle 2
FBE5 D009 4212 BNE BFBF0
FBE7 20ADFB 4213 JSR SFBAD ;invert output and schedule timer
FBEA D01D 4214 BNE BFC09 ;if output is zero again
FBEC E6A9 4215 INC ZA9 ;increment count for 2 cycles per bit
FBEF D019 4216 BNE BFC09
PBF0 20A6FB 4217 BFBF0 JSR SFBAB ;invert output and schedule timer
PBF3 D014 4218 BNE BFC09 ;if output is zero again
PBF5 A5A4 4219 LDA ZA4
PBF7 4901 4220 EOR $01 ;increment count for 2 cycles per bit
PBF9 85A4 4221 STA ZA4
PBFB F00F 4222 BEQ BFC0C ;if cycle is complete
Pbfd A5BD 4223 LDA ZBD
PBFf 4901 4224 EOR $01 ;invert data bit for next cycle
FC01 85BD 4225 STA ZBD
FC03 2901 4226 AND $01 ;add bit
FC05 459B 4227 EOR Z9B ;to parity
FC07 859B 4228 STA Z9B
FC09 4CBCFE 4229 BFC09 JMP JFEBC ;and exit IRQ
4230 ;
FC0C 46BD 4231 BFC0C LSR ZBD ;if 2 cycles done,
FC0E C6A3 4232 DEC ZA3 ;decrement bit count
FC10 A5A3 4233 LDA ZA3
FC12 F03A 4234 BEQ BFC4E ;after byte, send parity bit
FC14 10F3 4235 BPL BFC09 ;during byte, send next bit
FC16 2097FB 4236 BFC16 JSR SFB97 ;at end of byte, reset variables
FC19 58 4237 CLI ;allow IRQ
FC1A A5A5 4238 LDA ZA5 ;if block header count
FC1C F012 4239 BEQ BFC30 ;not zero
FC1E A200 4240 LDX $00
FC20 86D7 4241 STX ZD7 ;clear checksum
FC22 C6A5 4242 DEC ZA5 ;decrement header count
FC24 A6BE 4243 LDX ZBE
FC26 E002 4244 CPX $02 ;if phase 2
FC28 D002 4245 BNE BFC2C
FC2A 0980 4246 ORA $80 ;add 128 to count
FC2C 85BD 4247 BFC2C STA ZBD ;and send count as data for header
FC2E D0D9 4248 BNE BFC09 ;return from IRQ
FC30 20D1FC 4249 BFC30 JSR SFCD1 ;if end of data reached
FC33 900A 4250 BCC BFC3F ;do next byte
FC35 D091 4251 BNE BFBC8 ;if beyond, waste time
FC37 E6AD 4252 INC ZAD ;if at end, set pointer beyond end

```

FC39	A5D7	4253	LDA ZD7	;get checksum
FC3B	85BD	4254	STA ZBD	;and send it
FC3D	B0CA	4255	RTN	;return from IRQ
FC3F	A000	4256	BFC3F LDY \$00	
FC41	B1AC	4257	LDA (ZAC),Y	;get next byte to write
FC43	85BD	4258	STA ZBD	;and move to output character buffer
FC45	45D7	4259	EOR ZD7	;compute checksum
FC47	85D7	4260	STA ZD7	
FC49	20DBFC	4261	JSR SFCDB	;advance address
FC4C	D0BB	4262	BNE BFC09	;return from IRQ
FC4E	A59B	4263	BFC4E LDA Z9B	;get parity bit
FC50	4901	4264	EOR \$01	;invert it
FC52	85BD	4265	STA ZBD	;put into output buffer
FC54	4CBCFE	4266	BFC54 JMP JFEBC	;return from IRQ
		4267	;	
FC57	C6BE	4268	JFC57 DEC ZBE	;decrement phase
FC59	D003	4269	BNE BFC5E	;if phase is 0
FC5B	20CAFC	4270	JSR SFCCA	;stop cassette operations
FC5E	A950	4271	BFC5E LDA \$50	;set # cycles per sync to 40
FC60	85A7	4272	STA ZA7	
FC62	A208	4273	LDX \$08	
FC64	78	4274	SEI	;set IRQ vector
FC65	20BDFC	4275	JSR SFCBD	;for cassette write, part 1
FC68	DOEA	4276	BNE BFC54	;return from IRQ

```

4278 ;IRQ routine for cassette write, part 1
4279 ;
4280 ;entered for each block copy and for end of block
4281 ;
FC6A A978 4282 WFC6A LDA $78
FC6C 20AFFB 4283 JSR BFBAF ;invert output and schedule Timer
FC6F DOE3 4284 BNE BFC54 ;if output is zero again
FC71 C6A7 4285 DEC ZA7 ;decrement cycle count per sync
FC73 D0DF 4286 BNE BFC54 ;when zero,
FC75 2097FB 4287 JSR SFB97 ;reset cassette read/write variables
FC78 C6AB 4288 DEC ZAB ;decrement sync count per block
FC7A 10D8 4289 BPL BFC54 ;if negative,
FC7C A20A 4290 LDX $0A ;set IRQ vector
FC7E 20BDFC 4291 JSR SFCBD ;to other cassette write routine
FC81 58 4292 CLI ;enable IRQ
FC82 E6AB 4293 INC ZAB ;sync count set to 0
FC84 A5BE 4294 LDA ZBE ;if phase = 0
FC86 F030 4295 BEQ BFCB8 ;go terminate
FC88 208EFB 4296 JSR SFB8E ;move start of I/O area to current addr
FC8B A209 4297 LDX $09
FC8D 86A5 4298 STX ZA5 ;set block header count
FC8F 86B6 4299 STX ZB6 ;and current byte pointer
FC91 D083 4300 BNE BFC16 ;go write block
4301 ;
4302 ;switch from cassette IRQ to default IRQ
4303 ;
FC93 08 4304 SFC93 PHP ;save flags
FC94 78 4305 SEI ;disable IRQ
FC95 AD11D0 4306 LDA XD011
FC98 0910 4307 ORA $10 ;make screen visible again
FC9A 8D11D0 4308 STA XD011
FC9D 20CAFC 4309 JSR SFCCA ;stop cassette motor
FCA0 A97F 4310 LDA $7F
FCA2 8D0DDC 4311 STA XDC0D ;clear mask for all IRQ's in ICR1
FCA5 20DDFD 4312 JSR SFDDD ;reset TAL to 1/60 of a second
FCA8 ADA002 4313 LDA X02A0
FCAB F009 4314 BEQ BFCB6
FCAD 8D1503 4315 STA X0315 ;restore standard IRQ vector
FCB0 AD9F02 4316 LDA X029F
FCB3 8D1403 4317 STA X0314
FCB6 28 4318 BFCB6 PLP ;restore flags
FCB7 60 4319 RTS

```

```

4321 ;terminate cassette I/O
4322 ;
FCB8 2093FC 4323 BFCB8 JSR SFC93 ;switch from cassette to default IRQ
FCBB F097 4324 BEQ BFC54 ;return from IRQ
4325 ;
4326 ;set IRQ vector depending upon X.
4327 ;
FCBD BD93FD 4328 SFCBD LDA TFD9B-8,X ;move low byte of address
FCC0 8D1403 4329 STA X0314 ;into low byte of IRQ vector
FCC3 BD94FD 4330 LDA TFD9B-7,X ;then do high byte
FCC6 8D1503 4331 STA X0315
FCC9 60 4332 RTS
4333 ;
4334 ;stop cassette motor
4335 ;
FCCA A501 4336 SFCCA LDA Z01
FCCC 0920 4337 ORA $20 ;set bit 5 high in 6510 I/O register
FCCE 8501 4338 STA Z01 ;to stop cassette motor
FCD0 60 4339 RTS
4340 ;
4341 ;compare ZAC/D with ZAE/F
4342 ;
FCD1 38 4343 SFCD1 SEC
FCD2 A5AC 4344 LDA ZAC
FCD4 E5AE 4345 SBC ZAE ;compare low parts
FCD6 A5AD 4346 LDA ZAD
FCD8 E5AF 4347 SBC ZAF ;then high parts
FCDA 60 4348 RTS ;C is clear when ZAC/D is low
4349 ;
4350 ;increment ZAC/D
4351 ;
FCDB E6AC 4352 SFCDB INC ZAC ;increment low
FCDD D002 4353 BNE BFCE1
FCDF E6AD 4354 INC ZAD ;then high when necessary
FCE1 60 4355 BFCE1 RTS

```

```

4357 ;Reset routine
4358 ;
FCE2 A2FF 4359 WFCE2 LDX $FF
FCE4 78 4360 SEI ;disable IRQ
FCE5 9A 4361 TXS ;set stack pointer
FCE6 D8 4362 CLD ;clear decimal mode
FCE7 2002FD 4363 JSR SFD02 ;test for a cartridge
FCEA D003 4364 BNE BFCEF
FCEC 6C0080 4365 JMP (X8000) ;if found, execute cartridge Reset
4366 ;
FCEF 8E16D0 4367 BFCEF STX XD016 ;clear RES bit in video chip
FCF2 20A3FD 4368 JSR SFDA3 ;initialize I/O devices
FCF5 2050FD 4369 JSR SFD50 ;initialize memory pointers
FCF8 2015FD 4370 JSR SFD15 ;restore I/O vectors
FCFB 205BFF 4371 JSR JFF5B ;initialize screen and keyboard
FCFE 58 4372 CLI ;enable IRQ
FCFF 6C00A0 4373 JMP (XA000) ;begin execution
4374 ;
4375 ;check for the presence of a cartridge
4376 ;
FD02 A205 4377 SFD02 LDX $05 ;set length of key
FD04 BD0FFD 4378 BFD04 LDA TFD10-1,X ;compare table contents
FD07 DD0380 4379 CMP X8004-1,X ;to cartridge contents
FD0A D003 4380 BNE BFD0F ;exit with Z clear upon mismatch
FD0C CA 4381 DEX
FD0D D0F5 4382 BNE BFD04 ;repeat for 5 characters
FD0F 60 4383 BFD0F RTS ;Z set upon match
4384 ;
4385 ;cartridge key for autostart and warm start possibilities
4386 ;
FD10 C3C2CD 4387 TFD10 .BY "C+$80,"B+$80,"M+$80,"S,"0

```

```

4389 ;restore I/O vectors
4390 ;
FD15 A230 4391 SFD15 LDX <TFD30 ;set XY to standard table
FD17 A0FD 4392 LDY >TFD30
FD19 18 4393 CLC ;clear carry to use XY address
4394 ;
4395 ;set I/O vectors depending on XY
4396 ;
FD1A 86C3 4397 JFD1A STX ZC3 ;store XY in temporary pointer
FD1C 84C4 4398 STY ZC4 -
FD1E A01F 4399 LDY $1F ;set count for move
FD20 B91403 4400 BFD20 LDA X0314,Y
FD23 B002 4401 BCS BFD27 ;if carry set, don't fetch from XY addr
FD25 B1C3 4402 LDA (ZC3),Y ;move one byte
FD27 91C3 4403 BFD27 STA (ZC3),Y
FD29 991403 4404 STA X0314,Y ;to I/O vector
FD2C 88 4405 DEY
FD2D 10F1 4406 BPL BFD20 ;repeat for 32 bytes
FD2F 60 4407 RTS
4408 ;
4409 ;vectors for Operating System at $0314-0333
4410 ;
FD30 31EA 4411 TFD30 .W WEA31 ;IRQ
FD32 66FE 4412 .W WFE66 ;BRK
FD34 47FE 4413 .W WFE47 ;NMI
FD36 4AF3 4414 .W WF34A ;OPEN
FD38 91F2 4415 .W WF291 ;CLOSE
FD3A 0EF2 4416 .W WF20E ;set input device
FD3C 50F2 4417 .W WF250 ;set output device
FD3E 33F3 4418 .W WF333 ;restore I/O
FD40 57F1 4419 .W WF157 ;input
FD42 CAF1 4420 .W WF1CA ;output
FD44 EDF6 4421 .W WF6ED ;test Stop key
FD46 3EF1 4422 .W WF13E ;get
FD48 2FF3 4423 .W WF32F ;abort I/O
FD4A 66FE 4424 .W WFE66 ;unused (BRK)
FD4C A5F4 4425 .W WF4A5 ;load RAM
FD4E EDF5 4426 .W WF5ED ;save RAM

```



```

4428 ;initialize memory pointers
4429 ;
FD50 A900 4430 SFD50 LDA $00
FD52 A8 4431 TAY
FD53 990200 4432 BFD53 STA Z02,Y ;clear page 0
FD56 990002 4433 STA X0200,Y ;page 2
FD59 990003 4434 STA X0300,Y ;and page 3
FD5C C8 4435 INY
FD5D D0F4 4436 BNE BFD53
FD5F A23C 4437 LDX $3C
FD61 A003 4438 LDY $03
FD63 86B2 4439 STX ZB2 ;set tape buffer pointer
FD65 84B3 4440 STY ZB3
FD67 A8 4441 TAY
FD68 A903 4442 LDA $03
FD6A 85C2 4443 STA ZC2 ;initialize page for RAM test
FD6C E6C2 4444 BFD6C INC ZC2 ;continue RAM test on next memory page
FD6E B1C1 4445 BFD6E LDA (ZC1),Y
FD70 AA 4446 TAX ;save RAM contents
FD71 A955 4447 LDA $55
FD73 91C1 4448 STA (ZC1),Y
FD75 D1C1 4449 CMP (ZC1),Y
FD77 D00F 4450 BNE BFD88 ;if $55 not read back, then end of RAM
FD79 2A 4451 ROL A
FD7A 91C1 4452 STA (ZC1),Y
FD7C D1C1 4453 CMP (ZC1),Y
FD7E D008 4454 BNE BFD88 ;if $AA not read back, then end of RAM
FD80 8A 4455 TXA
FD81 91C1 4456 STA (ZC1),Y ;else restore original contents
FD83 C8 4457 INY ;bump address
FD84 D0E8 4458 BNE BFD6E ;continue on same page if not overflow
FD86 F0E4 4459 BEQ BFD6C ;else continue test on next page
FD88 98 4460 BFD88 TYA
FD89 AA 4461 TAX
FD8A A4C2 4462 LDY ZC2
FD8C 18 4463 CLC
FD8D 202DFE 4464 JSR JFE2D ;set top of memory pointer from X1
FD90 A908 4465 LDA $08
FD92 8D8202 4466 STA X0282 ;set bottom of memory pointer
FD95 A904 4467 LDA $04
FD97 8D8802 4468 STA X0288 ;set start of memory
FD9A 60 4469 RTS
4470 ;
4471 ;IRQ vectors
4472 ;
FD9B 6AFC 4473 TFD9B .W WFC6A ;X=$08, cassette write routine, part 1
FD9D CDFB 4474 .W WFBCD ;X=$0A cassette write routine, part 2
FD9F 31EA 4475 .W WEA31 ;X=$0C standard IRQ routine
FDA1 2CF9 4476 .W WF92C ;X=$0E cassette read routine

```

```

4478 ;initialize I/O devices
4479 ;
FDA3 A97F 4480 SFDA3 LDA $7F ;clear all mask bits in ICR1
FDA5 8D0DDC 4481 STA XDCOD
FDA8 8D0DDC 4482 STA XDDOD ;and in ICR2
FDAB 8D0DDC 4483 STA XDC00 ;set PA1 to $7F
FDAE A908 4484 LDA $08
FDB0 8D0EDC 4485 STA XDCOE ;one-shot mode in CRA1
FDB3 8D0EDD 4486 STA XDDOE ;and CRA2
FDB6 8D0FDC 4487 STA XDCOF ;and CRB1
FDB9 8D0FDD 4488 STA XDDOF ;and CRB2
FDBC A200 4489 LDX $00
FDBE 8E03DC 4490 STX XDC03 ;DDRB1 set to all inputs
FDC1 8E03DD 4491 STX XDD03 ;DDRB2 set to all inputs
FDC4 8E18D4 4492 STX XD418 ;kill volume in SID chip
FDC7 CA 4493 DEX
FDC8 8E02DC 4494 STX XDC02 ;DDRA1 set to all outputs
FDCB A907 4495 LDA $07
FDCD 8D00DD 4496 STA XDD00 ;PA2 bits 0-2 set high
FDD0 A93F 4497 LDA $3F
FDD2 8D02DD 4498 STA XDD02 ;DDRA2 bits 0-5 set to outputs
FDD5 A9E7 4499 LDA $E7
FDD7 8501 4500 STA Z01 ;initialize 6510 I/O register
FDD9 A92F 4501 LDA $2F
FDDB 8500 4502 STA Z00 ;and 6510 data direction register
4503 ;
4504 ;initialize TALL/TAH1 for 1/60 of a second
4505 ;
FDDD ADA602 4506 SFDDD LDA X02A6
FDE0 F00A 4507 BEQ BFDEC ;if this is an Intl. machine,
FDE2 A925 4508 LDA $25 ;use Intl value for 1/60 second delay
FDE4 8D04DC 4509 STA XDC04 ;and store in TALL
FDE7 A940 4510 LDA $40 ;high byte of value
FDE9 4CF3FD 4511 JMP JFDF3 ;go set high byte of timer
4512 ;
FDEC A995 4513 BFDEC LDA $95 ;since this is US machine, use US value
FDEE 8D04DC 4514 STA XDC04 ;for 1/60 second to set TALL
FDF1 A942 4515 LDA $42 ;high byte of value
FDF3 8D05DC 4516 JFDF3 STA XDC05 ;set TAH1
FDF6 4C6EFF 4517 JMP JFF6E ;go set CRA1 for continuous timer IRQ's
4518 ;
4519 ;initialize file name parameters
4520 ;
FDF9 85B7 4521 JFDF9 STA ZB7 ;set # of characters in file name
FDFB 86BB 4522 STX ZBB ;set address of file name
FDFD 84BC 4523 STY ZBC ;+ high byte
FDFF 60 4524 RTS
4525 ;
4526 ;initialize file parameters
4527 ;
FE00 85B8 4528 JFE00 STA ZB8 ;set logical file
FE02 86BA 4529 STX ZBA ;set current device
FE04 84B9 4530 STY ZB9 ;set secondary address
FE06 60 4531 RTS

```

```

        4533 ;read I/O status word
        4534 ;
FE07 A5BA 4535 JFE07 LDA ZBA
FE09 C902 4536      CMP $02      ;if RS-232 device,
FE0B D00D 4537      BNE BFE1A
FE0D AD9702 4538      LDA X0297      ;read RS-232 Status Register
FE10 48 4539      PHA
FE11 A900 4540      LDA $00
FE13 8D9702 4541      STA X0297      ;then clear RS-232 Status Register
FE16 68 4542      PLA      ;and restore original contents in A
FE17 60 4543      RTS
        4544 ;
        4545 ;control kernal messages
        4546 ;
FE18 859D 4547 JFE18 STA Z9D      ;set direct/run mode ($80/$00)
        4548 ;
        4549 ;read ST
        4550 ;
FE1A A590 4551 BFE1A LDA Z90      ;fetch current ST
        4552 ;
        4553 ;add A to current ST
        4554 ;
FE1C 0590 4555 SFE1C ORA Z90      ;add existing bits to A
FE1E 8590 4556      STA Z90      ;and store new ST
FE20 60 4557      RTS
        4558 ;
        4559 ;set timeout on serial bus
        4560 ;
FE21 8D8502 4561 JFE21 STA X0285      ;serial bus timeout flag (not used)
FE24 60 4562      RTS
        4563 ;
        4564 ;read/set top of memory
        4565 ;
FE25 9006 4566 JFE25 BCC JFE2D      ;if carry is clear, set pointers from XY
FE27 AE8302 4567 SFE27 LDX X0283      ;else fetch top of memory into XY
FE2A AC8402 4568      LDY X0284
FE2D 8E8302 4569 JFE2D STX X0283      ;set top of memory from X
FE30 8C8402 4570      STY X0284      ;and high byte from Y
FE33 60 4571      RTS
        4572 ;
        4573 ;read/set bottom of memory
        4574 ;
FE34 9006 4575 JFE34 BCC BFE3C      ;if carry is clear, set pointer from XY
FE36 AE8102 4576      LDX X0281      ;else fetch bottom of memory into XY
FE39 AC8202 4577      LDY X0282
FE3C 8E8102 4578 BFE3C STX X0281      ;set low byte bottom of memory from X
FE3F 8C8202 4579      STY X0282      ;and high byte from Y
FE42 60 4580      RTS

```

```

4582 ;NMI entry
4583 ;
FE43 78 4584 WFE43 SEI
FE44 6C1803 4585 JMP (X0318) ;perform NMI (normally FE47)
4586 ;
4587 ;standard NMI routine
4588 ;
FE47 48 4589 WFE47 PHA ;save A on stack
FE48 8A 4590 TXA
FE49 48 4591 PHA ;save X on stack
FE4A 98 4592 TYA
FE4B 48 4593 PHA ;save Y on stack
FE4C A97F 4594 LDA $7F
FE4E 8D0DDD 4595 STA XDD0D ;clear ICR2 IRQ mask bits
FE51 AC0DDD 4596 LDY XDD0D
FE54 301C 4597 BMI BFE72 ;if no IRQ's were present
FE56 2002FD 4598 JSR SFD02 ;check for a cartridge
FE59 D003 4599 BNE BFE5E ;if cartridge present,
FE5B 6C0280 4600 JMP (X8002) ;perform cartridge warm start
4601 ;
FE5E 20BCF6 4602 BFE5E JSR BF6BC ;scan the keyboard
FE61 20E1FF 4603 JSR SFFE1 ;check for Stop key
FE64 D00C 4604 BNE BFE72 ;and drop thru to BRK if depressed
4605 ;
4606 ;BRK routine
4607 ;
FE66 2015FD 4608 WFE66 JSR SFD15 ;restore I/O vectors
FE69 20A3FD 4609 JSR SFDA3 ;initialize I/O devices
FE6C 2018E5 4610 JSR SE518 ;initialize screen and keyboard
FE6F 6C02A0 4611 JMP (XA002) ;perform warm start

```

	4613	;internal NMI	
	4614	;	
FE72 98	4615	BFE72 TYA	;get ICR2
FE73 2DA102	4616	AND X02A1	;mask with ICR2 activity register
FE76 AA	4617	TAX	;save results
FE77 2901	4618	AND \$01	
FE79 F028	4619	BEQ BFEA3	;if transmitting
FE7B AD00DD	4620	LDA XDD00	
FE7E 29FB	4621	AND \$FB	;clear RS-232 output data line
FE80 05B5	4622	ORA ZB5	;and add bit to be transmitted
FE82 8D00DD	4623	STA XDD00	;and send it
FE85 ADA102	4624	LDA X02A1	;use ICR2 activity register
FE88 8D00DD	4625	STA XDD0D	;to set ICR2
FE8B 8A	4626	TXA	
FE8C 2912	4627	AND \$12	;if not receiving/waiting receiver edge,
FE8E F00D	4628	BEQ BFE9D	;send next bit on serial bus
FE90 2902	4629	AND \$02	
FE92 F006	4630	BEQ BFE9A	;if not receiving,
FE94 20D6FE	4631	JSR SFED6	;input next bit on RS-232
FE97 4C9DFE	4632	JMP BFE9D	;then send next bit on RS-232 bus
	4633	;	
FE9A 2007FF	4634	BFE9A JSR SFF07	;schedule TB2 using baud rate factor
FE9D 20BBEE	4635	BFE9D JSR SEEBB	;send next bit on RS-232
FEA0 4CB6FE	4636	JMP JFEB6	;exit from NMI
	4637	;	
FEA3 8A	4638	BFEA3 TXA	
FEA4 2902	4639	AND \$02	
FEA6 F006	4640	BEQ BFEEAE	;if receiving data,
FEA8 20D6FE	4641	JSR SFED6	;input next bit on RS-232, schedule TB2
FEAB 4CB6FE	4642	JMP JFEB6	;exit from NMI
	4643	;	
FEAE 8A	4644	BFEAE TXA	
FEAF 2910	4645	AND \$10	
FEB1 F003	4646	BEQ JFEB6	;if waiting for receiver edge,
FEB3 2007FF	4647	JSR SFF07	;schedule TB2 using baud rate factor
FEB6 ADA102	4648	JFEB6 LDA X02A1	;use ICR2 activity register
FEB9 8D00DD	4649	STA XDD0D	;to set ICR2
FEBB 68	4650	JFEBB PLA	
FEBD A8	4651	TAY	;restore Y
FEBE 68	4652	PLA	
FEBF AA	4653	TAX	;restore X
FEC0 68	4654	PLA	;restore A
FEC1 40	4655	RTI	;and return from NMI

```

4657 ;table of baud rate factors based upon
4658 ;((clock frequency/ baud rate/2)-100)
4659 ;this table applies to US machine clock frequency
4660 ;
FEC2 C127 4661 TFEC2 .W $27C1 ;50 baud
FEC4 3E1A 4662 .W $1A3E ;75
FEC6 C511 4663 .W $11C5 ;110
FEC8 740E 4664 .W $0E74 ;134.5
FECA EDOC 4665 .W $0CED ;150
FECC 4506 4666 .W $0645 ;300
FECE F002 4667 .W $02F0 ;600
FED0 4601 4668 .W $0146 ;1200
FED2 B800 4669 .W $00B8 ;1800
FED4 7100 4670 .W $0071 ;2400
4671 ;
4672 ;input next bit on RS-232 bus and schedule TB2
4673 ;
FED6 AD01DD 4674 SFED6 LDA XDD01
FED9 2901 4675 AND $01 ;mask bit read
FEDE 85A7 4676 STA ZA7 ;and save in temporary storage area
FEDD AD06DD 4677 LDA XDD06 ;use TAL2
FEE0 E91C 4678 SBC $1C ;and overhead
FEE2 6D9902 4679 ADC X0299 ;and baud rate full bit time
FEE5 8D06DD 4680 STA XDD06 ;to reset TBL2
FEE8 AD07DD 4681 LDA XDD07
FEEB 6D9A02 4682 ADC X029A
FEEE 8D07DD 4683 STA XDD07 ;and TBH2
FEF1 A911 4684 LDA $11
FEF3 8D0FDD 4685 STA XDD0F ;set force load + start TB2 bits in CRB2
FEF6 ADA102 4686 LDA X02A1
FEF9 8D0DDD 4687 STA XDD0D ;set ICR2 from ICR2 activity register
FEFC A9FF 4688 LDA $FF
FEFE 8D06DD 4689 STA XDD06 ;set TBL2
FF01 8D07DD 4690 STA XDD07 ;and TBH2 to maximum value
FF04 4C59EF 4691 JMP JEF59 ;add input bit to word being read
4692 ;
4693 ;schedule TB2 using baud rate factor
4694 ;
FF07 AD9502 4695 SFF07 LDA X0295 ;move bit time
FF0A 8D06DD 4696 STA XDD06 ;to TBL2
FF0D AD9602 4697 LDA X0296
FF10 8D07DD 4698 STA XDD07 ;and TBH2
FF13 A911 4699 LDA $11
FF15 8D0FDD 4700 STA XDD0F ;set force load and TB in CRB2
FF18 A912 4701 LDA $12
FF1A 4DA102 4702 EOR X02A1 ;invert waiting for receiver edge and
FF1D 8DA102 4703 STA X02A1 ;receiving data bits in activity reg
FF20 A9FF 4704 LDA $FF
FF22 8D06DD 4705 STA XDD06 ;set TBL2 to maximum value
FF25 8D07DD 4706 STA XDD07 ;set TBH2 to maximum value
FF28 AE9802 4707 LDX X0298
FF2B 86A8 4708 STX ZA8 ;set number of bits to send/receive
FF2D 60 4709 RTS

```

```

4711 ;continuation of baud rate calculation
4712 ;
FF2E AA 4713 SFF2E TAX
FF2F AD9602 4714 LDA X0296
FF32 2A 4715 ROL A ;multiply factor * 2
FF33 A8 4716 TAY
FF34 8A 4717 TXA
FF35 69C8 4718 ADC $C8 ;and add 200
FF37 8D9902 4719 STA X0299 ;to form baud rate
FF3A 98 4720 TYA
FF3B 6900 4721 ADC 0
FF3D 8D9A02 4722 STA X029A
FF40 60 4723 RTS
4724 ;
FF41 EA 4725 NOP
FF42 EA 4726 NOP
4727 ;
FF43 08 4728 JFF43 PHP
FF44 68 4729 PLA
FF45 29EF 4730 AND $EF ;clear BRK status bit
FF47 48 4731 PHA
4732 ;
4733 ;IRQ entry point
4734 ;
FF48 48 4735 WFF48 PHA ;save A on stack
FF49 8A 4736 TXA
FF4A 48 4737 PHA ;save X on stack
FF4B 98 4738 TYA
FF4C 48 4739 PHA ;save Y on stack
FF4D EA 4740 TSX
FF4E BD0401 4741 LDA X0100+4,X ;check BRK bit
FF51 2910 4742 AND $10
FF53 F003 4743 BEQ BFF58
FF55 6C1603 4744 JMP (X0316) ;perform BRK (normally FE66)
4745 ;
FF58 6C1403 4746 BFF58 JMP (X0314) ;else IRQ (normally EA31)
4747 ;
4748 ;addition to I/O device initialization
4749 ;
FF5B 2018E5 4750 JFF5B JSR SE518 ;initialize screen and keyboard
FF5E AD12D0 4751 BFF5E LDA XD012
FF61 D0FB 4752 BNE BFF5E ;wait for raster count to clear
FF63 AD19D0 4753 LDA XD019 ;use interrupt register
FF66 2901 4754 AND $01
FF68 8DA602 4755 STA X02A6 ;to set US/Intl machine flag (0=US)
FF6B 4CDDFD 4756 JMP SFDDDD ;set clock for 1/60 second IRQ's
4757 ;
4758 ;end of scheduling TA for 1/60 second IRQ's
4759 ;
FF6E A981 4760 JFF6E LDA $81
FF70 8D0DDC 4761 STA XDCOD ;set TA mask in ICR1
FF73 ADOEDC 4762 LDA XDCOE ;use CRA1
FF76 2980 4763 AND $80 ;bit 7 (50/60 Hz flag)
FF78 0911 4764 ORA $11 ;+ force load, continuous mode and TA1
FF7A 8DOEDC 4765 STA XDCOE ;to set CRA1
FF7D 4C8EEE 4766 JMP SEE8E ;set serial clock line high and return
FF80 00 4767 .BY $00 ;filler

```

```

4769 ;kernal vectors
4770 ;
FFB1 4C5BFF 4771      JMP JFF5B      ;initialize screen and keyboard
4772 ;
FFB4 4CA3FD 4773      JMP SFDA3      ;initialize I/O devices
4774 ;
FFB7 4C50FD 4775      JMP SFD50      ;initialize memory pointers
4776 ;
FFB8 4C15FD 4777      JMP SFD15      ;restore I/O vectors
4778 ;
FFBD 4C1AFD 4779      JMP JFD1A      ;set I/O vectors from XY
4780 ;
FF90 4C18FE 4781      JMP JFE18      ;control kernal messages
4782 ;
FF93 4CB9ED 4783      JMP SEDB9      ;send Secondary Address after Listen
4784 ;
FF96 4CC7ED 4785      JMP SEDC7      ;send Secondary Address after Talk
4786 ;
FF99 4C25FE 4787 SFF99 JMP JFE25      ;read/set top of memory
4788 ;
FF9C 4C34FE 4789 SFF9C JMP JFE34      ;read/set bottom of memory
4790 ;
FF9F 4C87EA 4791      JMP SEA87      ;scan keyboard
4792 ;
FFA2 4C21FE 4793      JMP JFE21      ;set timeout for serial bus
4794 ;
FFA5 4C13EE 4795      JMP JEE13      ;input byte on serial bus
4796 ;
FFA8 4CDDFD 4797      JMP JEDDD      ;output byte on serial bus
4798 ;
FFAB 4CEFD 4799      JMP SEDEF      ;send Untalk on serial bus
4800 ;
FFAE 4CFEED 4801      JMP SEDFE      ;send Unlisten on serial bus
4802 ;
FFB1 4C0CED 4803      JMP SEDOC      ;send Listen on serial bus
4804 ;
FFB4 4C09ED 4805      JMP SED09      ;send Talk on serial bus
4806 ;
FFB7 4C07FE 4807 SFFB7 JMP JFE07      ;read I/O status word
4808 ;
FFBA 4C00FE 4809 SFFBA JMP JFE00      ;set file parameters
4810 ;
FFBD 4CF9FD 4811 SFFBD JMP JFDF9      ;set file name parameters
4812 ;
FFC0 6C1A03 4813 SFFC0 JMP (X031A)      ;open a file (F34A)
4814 ;
FFC3 6C1C03 4815 SFFC3 JMP (X031C)      ;close a file (F291)
4816 ;
FFC6 6C1E03 4817 SFFC6 JMP (X031E)      ;set input device (F20E)
4818 ;
FFC9 6C2003 4819 SFFC9 JMP (X0320)      ;set output device (F250)
4820 ;
FFCC 6C2203 4821 SFFCC JMP (X0322)      ;restore I/O devices to default (F333)
4822 ;
FFCF 6C2403 4823 SFFCF JMP (X0324)      ;input char on current device (F157)
4824 ;
FFD2 6C2603 4825 SFFD2 JMP (X0326)      ;output char to current device (F1CA)
4826 ;
FFD5 4C9EF4 4827 SFFD5 JMP JF49E      ;load RAM from a device
4828 ;

```



```

FFD8 4CDDF5 4829 SFFD8 JMP JF5DD      ;save RAM to a device
      4830 ;
FFDB 4CE4F6 4831      JMP JF6E4      ;set real time clock
      4832 ;
FFDE 4CDDF6 4833      JMP JF6DD      ;read real time clock
      4834 ;
FFE1 6C2803 4835 SFFE1 JMP (X0328)    ;check Stop key (F6ED)
      4836 ;
FFE4 6C2A03 4837 SFFE4 JMP (X032A)    ;get a character (F13E)
      4838 ;
FFE7 6C2C03 4839      JMP (X032C)    ;close all channels and files (F32F)
      4840 ;
FFEA 4C9BF6 4841 SFFEA JMP JF69B      ;increment real time clock
      4842 ;
FFED 4C05E5 4843      JMP JE505      ;read organization of screen into XY
      4844 ;
FFF0 4C0AE5 4845      JMP JE50A      ;read/set XY cursor position
      4846 ;
FFF3 4C00E5 4847 SFFF3 JMP JE500      ;read base address of I/O devices
      4848 ;
      4849 ;garbage
      4850 ;
FFF6 525242 4851      .BY $52,$52,$42,$59
      4852 ;
      4853 ;6510 fixed vectors
      4854 ;
FFFA 43FE 4855      .W WFE43      ;NMI vector
FFFC E2FC 4856      .W WFCE2      ;RESET vector
FFFE 48FF 4857      .W WFF48      ;IRQ/BRK vector

```

BE00B	E00B	333					
BE00E	E00E	327					
BE01E	E01E	343					
BE06C	E06C	379					
BE070	E070	396					
BE07D	E07D	388					
BE0BE	E0BE	410					
BE0D3	E0D3	409					
BE0F9	E0F9	470	474	478	482	486	520 587
BE104	E104	459					
BE109	E109	465					
BE194	E194	546	586				
BE195	E195	539					
BE19E	E19E	543					
BE1A1	E1A1	554					
BE1B5	E1B5	560					
BE1D1	E1D1	537	578				
BE20D	E20D	618	627				
BE23F	E23F	647					
BE29D	E29D	688	727				
BE2A0	E2A0	691					
BE2AD	E2AD	700					
BE316	E316	756					
BE324	E324	761					
BE337	E337	770					
BE33D	E33D	775					
BE386	E386	833					
BE391	E391	821					
BE3A8	E3A8	838					
BE3B9	E3B9	842					
BE3E2	E3E2	876					
BE421	E421	902					
BE455	E455	941					
BE4B6	E4B6	960					
BE4E2	E4E2	985					
BE4EB	E4EB	983					
BE513	E513	1016					
BE54D	E54D	1056					
BE555	E555	1052					
BE560	E560	1062					
BE570	E570	1077					
BE57C	E57C	1072					
BE58D	E58D	1091					
BE597	E597	1087					
BE5AA	E5AA	1108					
BE5B9	E5B9	1119					
BE5CA	E5CA	1154					
BE5CD	E5CD	1132	1151	1187			
BE5E7	E5E7	1135					
BE5F3	E5F3	1150					
BE5FE	E5FE	1143					
BE606	E606	1161					
BE60F	E60F	1159					
BE63A	E63A	1169	1173	1177			
BE64A	E64A	1197					
BE650	E650	1199					
BE654	E654	1201	1202				
BE65D	E65D	1178					
BE66F	E66F	1213					

BE672	E672	1216
BE674	E674	1207
BE682	E682	1226
BE690	E690	1234
BE699	E699	1245
BE69F	E69F	1248
BE6A8	E6A8	1317 1407 1425 1475 1506 1509 1514 1541 2007 2019
BE6B0	E6B0	1259
BE6CD	E6CD	1278
BE6DA	E6DA	1283 1674 1708
BE6F4	E6F4	1299
BE6F7	E6F7	1276
BE700	E700	1274
BE70B	E70B	1313
BE72A	E72A	1337
BE731	E731	1341
BE73D	E73D	1347
BE73F	E73F	1349
BE745	E745	1345
BE74C	E74C	1355
BE759	E759	1361
BE762	E762	1379
BE77E	E77E	1359
BE785	E785	1386
BE78B	E78B	1390
BE792	E792	1393
BE7A8	E7A8	1417 1418
BE7AA	E7AA	1402
BE7AD	E7AD	1396
BE7C0	E7C0	1423
BE7C8	E7C8	1421
BE7CB	E7CB	1384
BE7CE	E7CE	1410
BE7DC	E7DC	1434
BE7E3	E7E3	1437
BE7EA	E7EA	1441
BE7FE	E7FE	1451
BE805	E805	1453
BE80A	E80A	1469
BE826	E826	1455
BE829	E829	1447
BE82D	E82D	1445
BE832	E832	1478
BE847	E847	1490
BE84C	E84C	1483
BE854	E854	1496
BE864	E864	1502
BE86A	E86A	1500
BE871	E871	1485 1492 1494
BE874	E874	1512
BE880	E880	1529
BE888	E888	1526
BE8A5	E8A5	1552
BE8B0	E8B0	1548
BE8B7	E8B7	1567
BE8C2	E8C2	1563
BE8CA	E8CA	1572
BE8CD	E8CD	1582
BE8D6	E8D6	1580
BE8F6	E8F6	1630

BE8FF	E8FF	1614
BE913	E913	1609
BE918	E918	1625
BE922	E922	1620
BE94D	E94D	1645 1647
BE956	E956	1641
BE981	E981	1668 1669
BE98F	E98F	1694
BE9A6	E9A6	1688 1689
BE9AB	E9AB	1706
BE9BA	E9BA	1702
BE9BF	E9BF	1698
BE9D4	E9D4	1723
BEA07	EA07	1757
BEA5C	EA5C	1799
BEA61	EA61	840 1790 1792
BEA71	EA71	1811
BEA79	EA79	1816
BEA7B	EA7B	1818
BEAA8	EAA8	1875
BEAAB	EAA8	1852
BEAB3	EAB3	1870
BEAC9	EAC9	1858 1860
BEACB	EACB	1863
BEACC	EACC	1854
BEADC	EADC	1868
BEAFO	EAF0	1883
BEAFB	EAFB	1840
BEB0D	EB0D	1890 1895 1897 1899
BEB17	EB17	1903
BEB26	EB26	1886 1893
BEB42	EB42	1891 1901 1905 1907 1912 1918 1922 1934
BEB64	EB64	1932
BEB6B	EB6B	1946
BEB76	EB76	1936 1940
BEC58	EC58	2001
BEC5B	EC5B	2015
BEC5E	EC5E	2003
BEC69	EC69	2010
BEC72	EC72	2013
BED20	ED20	2104
BED2E	ED2E	2115
BED50	ED50	2135
BED55	ED55	2137
BED5A	ED5A	2133 2139
BED66	ED66	2145 2163
BED7A	ED7A	2149
BED7D	ED7D	2151
BED9F	ED9F	2173
BEDAD	EDAD	2130
BEDB0	EDB0	2147 2171
BEDD6	EDD6	2203
BEDE6	EDE6	2210
BEDEB	EDEB	2213
BEE03	EE03	2183
BEE09	EE09	2239
BEE1B	EE1B	2251
BEE20	EE20	2274
BEE30	EE30	2262
BEE3E	EE3E	2260

BEE47	EE47	2265					
BEE56	EE56	2263					
BEE5A	EE5A	2279	2281	2289			
BEE67	EE67	2285	2287				
BEE80	EE80	2292					
BEEB6	EEB6	2342					
BEEC8	EEC8	2353					
BEED1	EED1	2375	2377	2387			
BEED7	EED7	2359					
EEEE6	EEE6	2379	2382	2384			
EEEE7	EEE7	2371	2381	2383			
BEEF2	EEF2	2367					
BEF6	EEF6	2369					
BEFC	EEFC	2368					
BEF00	EF00	2350					
BEF06	EF06	2349	2565				
BEF13	EF13	2390					
BEF2E	EF2E	2392					
BEF31	EF31	2393					
BEF39	EF39	2401					
BEF54	EF54	2427					
BEF58	EF58	2429					
BEF6D	EF6D	2457	2495	2499	2501		
BEF6E	EF6E	2494					
BEF70	EF70	2440					
BEF7E	EF7E	2472	2511				
BEF90	EF90	2437					
BEF97	EF97	2439					
BEFA9	EFA9	2490					
BEFB1	EFB1	2487					
BEFC5	EFC5	2498					
BEFCA	EPCA	2481					
BEFCD	EFCD	2515					
BEFDO	EFDO	2514					
BEFDB	EFDB	2452					
BEFF2	EFF2	2529					
BEFF9	EFF9	2531					
BF006	F006	2537					
BF00D	F00D	2525	2580	3135			
BF012	F012	2522	2526	2536			
BF014	F014	2549					
BF04C	F04C	2556					
BF062	F062	2584					
BF070	F070	2590					
BF077	F077	2597					
BF07D	F07D	2575	2577				
BF084	F084	2581					
BF09C	F09C	2606					
BF0AA	F0AA	2625					
BF0BB	F0BB	2622					
BF12F	F12F	2658	3320	3324	3445	3534	3565 3695 3700
BF13C	F13C	2651					
BF14A	F14A	2665					
BF155	F155	2667					
BF166	F166	2672	2682				
BF173	F173	2690					
BF18D	F18D	2705					
BF193	F193	2704					
BF196	F196	2701					
BF1A9	F1A9	2722					

BF1AD	F1AD	2696	
BF1B1	F1B1	2750	
BF1B3	F1B3	2747	
BF1B4	F1B4	2724	2745
BF1B5	F1B5	2733	
BF1B8	F1B8	2698	2751
BF1D5	F1D5	2758	
BF1DB	F1DB	2762	
BF1F8	F1F8	2775	
BF1FD	F1FD	2777	
BF207	F207	2791	
BF208	F208	2773	
BF216	F216	2801	
BF22A	F22A	2811	
BF233	F233	2806	2808 2816 2835
BF237	F237	2809	
BF245	F245	2828	
BF258	F258	2841	
BF25F	F25F	2858	
BF262	F262	2846	
BF26F	F26F	2853	
BF275	F275	2850	2874
BF279	F279	2851	
BF286	F286	2868	
BF289	F289	2870	
BF298	F298	2880	
BF2BA	F2BA	2899	
BF2BF	F2BF	2902	
BF2C8	F2C8	2893	
BF2E0	F2E0	2919	
BF2EE	F2EE	2891	
BF2F1	F2F1	2888	2890 2913 2926 2929
BF30D	F30D	2941	
BF316	F316	2961	
BF32E	F32E	2959	
BF33C	F33C	2983	
BF343	F343	2986	
BF351	F351	2996	
BF359	F359	3000	
BF362	F362	3005	
BF384	F384	3020	
BF38B	F38B	3024	
BF393	F393	3030	
BF3AC	F3AC	3049	
BF3AF	F3AF	3040	
BF3B8	F3B8	3035	
BF3C2	F3C2	3042	3048
BF3D1	F3D1	3060	
BF3D3	F3D3	3017	3019 3022 3072 3074
BF3D4	F3D4	3037	3043 3047 3054
BF3F6	F3F6	3083	
BF3FC	F3FC	3095	
BF406	F406	3089	
BF40F	F40F	3108	
BF41D	F41D	3103	
BF43A	F43A	3117	
BF446	F446	3113	
BF45C	F45C	3131	3134
BF474	F474	3143	
BF4AF	F4AF	3186	

BF4B2	F4B2	3182	
BF4BF	F4BF	3189	
BF4F0	F4F0	3210	
BF4F3	F4F3	3228	3243
BF501	F501	3220	
BF51C	F51C	3231	
BF51E	F51E	3234	
BF524	F524	3240	
BF530	F530	3206	3265 3268
BF533	F533	3187	
BF539	F539	3250	
BF541	F541	3254	
BF549	F549	3276	
BF556	F556	3261	
BF55D	F55D	3263	
BF56C	F56C	3285	
BF579	F579	3274	
BF57D	F57D	3283	
BF5A9	F5A9	3246	
BF5AE	F5AE	3258	3264 3267 3272
BF5C7	F5C7	3332	
BF5D1	F5D1	3318	3322 3326
BF5DA	F5DA	3339	
BF5F1	F5F1	3361	3417
BF5F4	F5F4	3357	
BF605	F605	3366	
BF624	F624	3393	
BF63A	F63A	3386	
BF63F	F63F	3382	
BF657	F657	3399	
BF659	F659	3362	
BF65F	F65F	3411	
BF676	F676	3424	
BF68D	F68D	3433	
BF68E	F68E	3419	3428 3430 3443
BF6A7	F6A7	3452	3454
BF6BC	F6BC	3463	3469 3787 4602
BF6CC	F6CC	3476	
BF6DA	F6DA	3471	
BF6DC	F6DC	3479	
BF6FA	F6FA	3503	
BF729	F729	3533	
BF74B	F74B	3556	3558
BF757	F757	3571	
BF761	F761		
BF767	F767	3563	
BF769	F769	3550	3554
BF781	F781	3596	
BF7A5	F7A5	3623	
BF7B7	F7B7	3617	
BF7CF	F7CF	3583	
BF7F7	F7F7	3678	
BF80B	F80B	3670	
BF80C	F80C	3664	
BF81E	F81E	3716	
BF821	F821	3698	
BF836	F836	3693	3706 3714
BF86E	F86E	3725	
BF875	F875	3736	
BF8B5	F8B5	3780	

BF8E7	F8E7	3778
BF8DC	F8DC	3744 3785
BF8E1	F8E1	3794
BF8F7	F8F7	3816
BF8FE	F8FE	3825
BF92A	F92A	3836 3837 3839
BF92C	F92C	3853
BF969	F969	3875
BF988	F988	3879 3925
BF98B	F98B	3893 3929 3943
BF993	F993	3884
BF997	F997	3889 3969
BF999	F999	3901
BF9AC	F9AC	3873 3897 3899 3945 3947 3949
BF9BC	F9BC	3917
BF9C9	F9C9	3956
BF9D2	F9D2	3914 3919 3954
BF9D5	F9D5	3911
BF9DE	F9DE	3934
BF9E0	F9E0	3933
BF9F7	F9F7	3924 3941
BFA18	FA18	3964
BFA1F	FA1F	3966 3968
BFA44	FA44	3981
BFA53	FA53	3992
BFA5D	FA5D	3983
BFA70	FA70	4011
BFA86	FA86	4017
BFA8A	FABA	4020 4023 4031 4033 4044 4050 4053
BFA8D	FA8D	4015
BFAA3	FAA3	4037
BFAA9	FAA9	4029
BFABA	FABA	4038 4040
BFACO	FACO	4028
BFACE	FACE	4055
BFAD6	FAD6	4062
BFAEB	FAEB	4069 4073
BFBO8	FBO8	4067
BFB2F	FB2F	4103
BFB33	FB33	4080
BFB3A	FB3A	4077 4089 4111
BFB43	FB43	4093 4096 4099 4107 4114 4116
BFB5C	FB5C	4130
BFB68	FB68	4133
BFB72	FB72	4147
BFB8B	FB8B	4121 4135 4137 4150
BFAF	FBAF	4179 4283
BFB08	FBC8	4251
BFB03	FBE3	4201
BFBFO	FBFO	4212
BFC09	FC09	4199 4205 4208 4214 4216 4218 4235 4248 4255 4262
BFC0C	FC0C	4222
BFC16	FC16	4300
BFC2C	FC2C	4245
BFC30	FC30	4239
BFC3F	FC3F	4250
BFC4E	FC4E	4234
BFC54	FC54	4276 4284 4286 4289 4324
BFC5E	FC5E	4269
BFCB6	FCB6	4314

BFCB8	FCB8	4295	
BFCE1	FCE1	4353	
BFCEF	FCEF	4364	
BFD04	FD04	4382	
BFD0F	FD0F	4380	
BFD20	FD20	4406	
BFD27	FD27	4401	
BFD53	FD53	4436	
BFD6C	FD6C	4459	
BFD6E	FD6E	4458	
BFD88	FD88	4450	4454
BFDEC	FDEC	4507	
BFE1A	FE1A	4537	
BFE3C	FE3C	4575	
BFE5E	FE5E	4599	
BFE72	FE72	4597	4604
BFE9A	FE9A	4630	
BFE9D	FE9D	4628	4632
BFEA3	FEA3	4619	
BFEAE	FEAE	4640	
BFF58	FF58	4743	
BFF5E	FF5E	4752	
JE043	E043	704	767
JE0E3	EOE3	426	
JE500	E500	4847	
JE505	E505	4843	
JE50A	E50A	4845	
JE566	E566	1096	1394
JE632	E632	2687	2694
JE691	E691	1438	
JE693	E693	1352	
JE697	E697	1356	1387 1480
JE773	E773	1363	
JE7D4	E7D4	1338	
JE891	E891	1342	1442
JE958	E958	1709	
JE967	E967	1279	1665
JEAE0	EAE0	1953	
JEC44	EC44	1428	
JEC4F	EC4F	1518	1998
JEDB2	EDB2	2267	
JEDDD	EDDD	2764	3092 3378 3380 3384 4797
JEE13	EE13	2740	3201 3207 3223 4795
JEF3B	EF3B	2467	2564 2593
JEF59	EF59	4691	
JEFE1	EFE1	2854	
JF04D	F04D	2812	
JF12B	F12B	3341	
JF1FC	F1FC	2796	
JF248	F248	2830	
JF409	F409	3025	
JF440	F440	3120	
JF47D	F47D	2907	3148
JF49E	F49E	4827	
JF5C1	F5C1	3446	
JF5DD	F5DD	4829	
JF633	F633	3221	
JF654	F654	3096	
JF69B	F69B	4841	
JF6DD	F6DD	4833	

JF6E4	F6E4	4831				
JF6FB	F6FB	3006				
JF6FE	F6FE	3001				
JF701	F701	2802	2842			
JF704	F704	3044	3247			
JF707	F707	2836	2875	3086		
JF70A	F70A	2817	2997			
JF70D	F70D	2847				
JF710	F710	3190	3367			
JF713	F713	3031	3183	3251	3255	3358 3412
JF8BE	F8BE	3788				
JFA10	FA10	3894				
JFA60	FA60	3876				
JFB48	FB48	4063				
JFB4A	FB4A	4059				
JFC57	FC57	4209				
JFD1A	FD1A	4779				
JFDF3	FD3	4511				
JFDF9	FD9	4811				
JFE00	FE00	4809				
JFE07	FE07	4807				
JFE18	FE18	4781				
JFE21	FE21	4793				
JFE25	FE25	4787				
JFE2D	FE2D	3154	4464	4566		
JFE34	FE34	4789				
JFE86	FE86	4636	4642	4646		
JFEB3	FEB3	3930	3961	4002	4026	4153 4229 4266
JFF43	FF43	3841				
JFF5B	FF5B	4371	4771			
JFF6E	FF6E	4517				
SE059	E059	350				
SE05D	E05D	364				
SE0F6	E0F6	714				
SE1D4	E1D4	515	532			
SE200	E200	599	604	642	651	
SE206	E206	596	598	603	641	650 657
SE20E	E20E	612	658			
SE211	E211	634				
SE219	E219	576	583			
SE257	E257	597				
SE26B	E26B	711				
SE2DC	E2DC	721				
SE3BF	E3BF	829				
SE422	E422	830				
SE453	E453	828				
SE4AD	E4AD	477				
SE4DA	E4DA	1754				
SE4E0	E4E0	3573				
SE518	E518	4610	4750			
SE544	E544	1513				
SE56C	E56C	1019	1320	1493	1531	
SE5A0	E5A0	1026	1095			
SE5B4	E5B4	1141	2669			
SE684	E684	1205	1351			
SE6B6	E6B6	1252				
SE6ED	E6ED	1171	1301			
SE701	E701	1362	1508			
SE716	E716	1128	1217	2760		
SE87C	E87C	1305	1404	1424	1540	

SE8A1	E8A1	1365	1503				
SE8B3	E8B3	1270	1398				
SE8CB	E8CB	1427	1517				
SE8EA	E8EA	1284	1527	1670			
SE965	E965	1456					
SE9C8	E9C8	1613	1693				
SE9E0	E9E0	1716					
SE9F0	E9F0	1302	1607	1686	1750		
SE9FF	E9FF	1060	1615	1695			
SEA13	EA13	1140	1251				
SEA1C	EA1C	1808					
SEA24	EA24	1368	1458	1728	1751	1765	1802
SEA87	EA87	1822	4791				
SED09	ED09	2826	3198	4805			
SED0C	ED0C	2866	3078	3372	3401	4803	
SED11	ED11	2234					
SED36	ED36	2188	2197				
SED40	ED40	2107	2215				
SEDB9	EDB9	2871	3081	3374	3405	4783	
SEDBE	EDBE	2200	2235	2869			
SEDC7	EDC7	2832	3200	4785			
SEDC	EDCC	2829					
SEDEF	EDEF	2987	3244	4799			
SEDFE	EDFE	2984	3394	3406	4801		
SEE06	EE06	2293					
SEE85	EE85	2116	2131	2153	2201	2241	2249 2270
SEE8E	EE8E	2121	2140	2224	4766		
SEE97	EE97	2113	2122	2128	2152	2242	2256
SEEA0	EEA0	2150	2199	2269	2290		
SEEA9	EEA9	2129	2134	2136	2138	2172	2202 2250 2261 2333
SEEB3	EEB3	2123					
SEEBB	EEBB	4635					
SEF4A	EF4A	3109					
SF017	F017	2795					
SF028	F028	2545					
SF086	F086	2674					
SFOA4	FOA4	2101	3759				
SF14E	F14E	2744					
SF199	F199	2700	2703	2727			
SF1DD	F1DD	2917					
SF2F2	F2F2	2895					
SF30F	F30F	2800	2840	2999			
SF314	F314	2879					
SF31F	F31F	2804	2844	2884			
SF3D5	F3D5	3021	3196	3369			
SF483	F483	2896	3100				
SF5AF	F5AF	3038	3193	3259			
SF5D2	F5D2	3215	3307				
SF642	F642	2933	3245	3387			
SF68F	F68F	3370	3420				
SF72C	F72C	3046	3266	3560	3663		
SF76A	F76A	2928	3056	3427	3435		
SF7D0	F7D0	2914	3029	3253	3416	3582	3649 3684
SF7D7	F7D7	3624	3723	3740			
SF7EA	F7EA	3041	3262	3674			
SF80D	F80D	2721	2774				
SF817	F817	3036	3257	3724			
SF82E	F82E	3692	3697	3713			
SF838	F838	3053	3418	3743			
SF841	F841	2723	3547				

SF84A	F84A	3308					
SF864	F864	2776	2918				
SF867	F867	3429					
SF86B	F86B	3627					
SF8D0	F8D0	3696	3786				
SF8E2	F8E2	3927	3960	3978	4009		
SF88E	F88E	3376	4047	4139	4296		
SFB97	F897	3770	4006	4236	4287		
SFBA6	FBA6	4217					
SFBAD	FBA6	4213					
SFBB1	FBB1	4204					
SFC93	FC93	3795	4138	4323			
SFCBD	FCBD	3767	4275	4291			
SFCCA	FCCA	4270	4309				
SFCD1	FCD1	3381	4061	4146	4249		
SFCDB	FCDB	3392	4120	4145	4261		
SFD02	FD02	4363	4598				
SFD15	FD15	4370	4608	4777			
SFD50	FD50	4369	4775				
SFDA3	FDA3	4368	4609	4773			
SFDDD	FDDD	4312	4756				
SFE1C	FE1C	2180	2272	2707	3236	4022	4057 4113 4152
SFE27	FE27	2897	3141				
SFED6	FED6	4631	4641				
SFF07	FF07	4634	4647				
SFF2E	FF2E	3128					
SFF99	FF99	893					
SFF9C	FF9C	889					
SFFB7	FFB7	541	552				
SFFBA	FFBA	595	602	608	640	649	656
SFFBD	FFBD	592	633	663			
SFFC0	FFC0	577					
SFFC3	FFC3	585					
SFFC6	FFC6	481					
SFFC9	FFC9	957					
SFFCC	FFCC	810	3505	3530			
SFFCF	FFCF	473					
SFFD2	FFD2	469	2655	3329	3538	3568	
SFFD5	FFD5	536					
SFFD8	FFD8	519					
SFFE1	FFE1	3219	3385	3792	4603		
SFFE4	FFE4	485					
SFFEA	FFEA	1788					
SFFP3	FFP3	411					
TA364	A364	547	548				
TA376	A376	563	564				
TB9BC	B9BC	762	763				
TBFC4	BFC4	348	349				
TE08D	E08D	431	432				
TE092	E092	434	435				
TE2E0	E2E0	667	668	771	772		
TE2E5	E2E5	674	675				
TE2EA	E2EA	683	684	696	697		
TE2EF	E2EF	702	703				
TE33E	E33E	765	766				
TE3A2	E3A2	844	873				
TE447	E447	938					
TE460	E460	921	922				
TE473	E473	911	912				
TE4EC	E4EC	3122	3123				

TE8DA	E8DA	1579	
TEB79	EB79	1949	1951
TEB81	EB81	1842	1844 1957
TEBC2	ECB2	1958	
TEC03	EC03	1959	
TEC78	EC78	1960	
TECB9	ECB9	1105	
TECE7	ECE7	1147	
TECF0	ECF0	1082	1610 1690 1739
TF0BD	F0BD	2652	
TF0C9	F0C9		
TF0D4	F0D4		
TF0D8	F0D8		
TF0EB	F0EB		
TF106	F106		
TF10E	F10E		
TF116	F116		
TF120	F120		
TF127	F127		
TFD10	FD10	4378	
TFD30	FD30	4391	4392
TFD9B	FD9B	4328	4330
TFEC2	FEC2	3118	3119
WA483	A483	929	
WA57C	A57C	930	
WA71A	A71A	931	
WA7E4	A7E4	932	
WAE86	AE86	933	
WB1AA	B1AA	868	869
WB248	B248	860	861
WB391	B391	864	865
WE097	E097		
WE12A	E12A		
WE147	E147	493	495
WE156	E156		
WE165	E165		
WE168	E168		
WE1BE	E1BE		
WE1C7	E1C7		
WE264	E264		
WE2B4	E2B4		
WE30E	E30E		
WE38B	E38B	928	
WEA31	EA31	4411	4475
WEB48	EB48	1030	1032
WF13E	F13E	4422	
WF157	F157	4419	
WF1CA	F1CA	4420	
WF20E	F20E	4416	
WF250	F250	4417	
WF291	F291	4415	
WF32F	F32F	4423	
WF333	F333	4418	
WF34A	F34A	4414	
WF4A5	F4A5	4425	
WF5ED	F5ED	4426	
WF6ED	F6ED	4421	
WF92C	F92C	4476	
WF8CD	F8CD	4474	
WFC6A	FC6A	4473	

X0311	0311	862			
X0312	0312	863			
X0314	0314	3763	4317	4329	4400 4404 4746
X0315	0315	3765	3783	4315	4331
X0316	0316	4744			
X0318	0318	4585			
X031A	031A	4813			
X031C	031C	4815			
X031E	031E	4817			
X0320	0320	4819			
X0322	0322	4821			
X0324	0324	4823			
X0326	0326	4825			
X0328	0328	4835			
X032A	032A	4837			
X032C	032C	4839			
X0330	0330	3174			
X0332	0332	3352			
X8000	8000	4365			
X8002	8002	4600			
X8004	8004	4379			
XA000	A000	4373			
XA002	A002	4611			
XA408	A408	910			
XA437	A437	467	556		
XA43A	A43A	822			
XA474	A474	824			
XA52A	A52A	566			
XA533	A533	571			
XA644	A644	924			
XA663	A663	462			
XA677	A677	572			
XA67A	A67A	813			
XA68E	A68E	570			
XAB1E	AB1E	549	565	913	923
XAD8A	AD8A	491			
XAD9E	AD9E	659			
XAEFD	AEFD	625			
XAF08	AF08	628			
XB6A3	B6A3	660			
XB79E	B79E	613	635		
XB7F7	B7F7	492			
XB849	B849	689			
XB850	B850	685	773		
XB853	B853	346	682		
XB867	B867	392	436	669	698
XB8D7	B8D7	451			
XBA28	BA28	363	367	383	433
XBA89	BA89	354			
XBAD4	BAD4	328			
XB807	B807	677			
XB80F	B80F	724	764		
XBBA2	BBA2	430	717		
XBBC7	BBC7	373			
XBBCA	BBCA	361	708		
XBBD4	BBD4	454			
XBC0C	BC0C	673	678		
XBC0F	BC0F	324			
XBC2B	BC2B	408			
XBCCC	BCCC	329	679		

[illegible]

Z49	0049	584	601	607	636	645	655		
Z4A	004A	643	654						
Z4E	004E	712	713	722	723				
Z53	0053	878							
Z54	0054	858							
Z56	0056	323	344						
Z57	0057	362	365	366	715	716			
Z5C	005C	393	394						
Z61	0061	325	339	340	447	450	758		
Z62	0062	416	438	440					
Z63	0063	422	441	443					
Z64	0064	419	442	444					
Z65	0065	425	437	439					
Z66	0066	446	686	690	719	754			
Z67	0067	375	395						
Z68	0068	880							
Z69	0069	338	341						
Z6E	006E	676							
Z6F	006F	352	681						
Z70	0070	345	448						
Z71	0071	359	371	374	376	381	384	390	
Z72	0072	360	372	380	382	385	391		
Z73	0073	874							
Z7A	007A	544	837						
Z78	007B	558	839						
Z8B	008B	428	429	452	453				
Z90	0090	2291	2732	2834	2873	2955	3076	3082	3180
		3242	3269	3721	4551	4555	4556		
Z91	0091	981	3480	3501					
Z92	0092	3906	3907	3932	3939				
Z93	0093	3178	3230	3338	3545	3549	3722	4068	4102
Z94	0094	2103	2108	2209	2212				
Z95	0095	2111	2148	2187	2196	2217			
Z96	0096	3948	3963	3982	3986	3990			
Z97	0097	2673	2675	2699	2709	2716			
Z98	0098	2939	2940	2942	2957	2977	3003	3008	
Z99	0099	1103	1211	2572	2664	2681	2819	2985	2990
Z9A	009A	1101	1214	2519	2756	2859	2982	2988	
Z9B	009B	3928	3951	3952	4170	4227	4228	4263	
Z9C	009C	3733	3874	3979	4007				
Z9D	009D	2650	3317	3442	3532	3562	4547		
Z9E	009E	2552	2768	2783	2790	3581	3597	3614	3615
		3731	4079	4081	4088	4092	4134		
Z9F	009F	3612	3619	3622	3666	3672	3676	3732	4091
ZA0	00A0	3455	3461	3464	3488	3495			
ZA1	00A1	984	3453	3459	3465	3487	3494	3572	
ZA2	00A2	3451	3457	3466	3486	3493			
ZA3	00A3	2106	2109	2132	3878	3923	3955	3967	4166
ZA4	00A4	2282	2294	3908	3910	3921	4168	4219	4221
ZA5	00A5	2142	2162	2248	2264	2273	2276	2288	4238
ZA6	00A6	2708	2726	2782	3065	3685	3686		
ZA7	00A7	2441	2444	2451	2471	2496	4012	4034	4065
ZA8	00A8	2438	2450	2456	3898	3984	3999	4169	4200
ZA9	00A9	2436	2465	2473	3900	3902	3944	4000	4171
ZAA	00AA	2445	2484	2513	3728	4014	4025	4042	4043
ZAB	00AB	2442	2443	2497	3626	3742	4049	4141	4143
ZAC	00AC	1594	1611	1657	1676	1691	1718	1729	3377
		4106	4119	4142	4160	4257	4344	4352	
ZAD	00AD	1596	1655	1678	1715	1731	3379	4082	4097
ZAE	00AE	1598	1653	1680	1720	1730	3202	3212	3233
									3238
									3239
									3299
									3311

	3345 3590 3606 3630 3654 4345
ZAF	00AF 1500 1651 1682 1734 3208 3214 3241 3302 3312 3346 3588 3609 3632 3658 4347
ZB0	00B0 3730 3806 3810 3815 3869 3882 3887 3891 3935 3937 3975
ZB1	00B1 3805 3812 3813 3818 3820 3826 3854 3863 3864 3866 3868 3872 3883 3888 3892 3905 3971 3974
ZB2	00B2 2728 2780 2784 3063 3278 3281 3287 3289 3292 3294 3552 3567 3594 3598 3601 3604 3607 3610 3620 3642 3673 4439
ZB3	00B3 3643 4440
ZB4	00B4 2348 2358 2373 2376 2378 2385 2398 3729 3896 3913 3953 3965 3980 3989 3994
ZB5	00B5 2362 2396 3991 4016 4030 4054 4622
ZB6	00B6 2351 2403 4001 4032 4075 4076 4109 4110 4198 4207 4299
ZB7	00B7 3039 3073 3088 3094 3102 3188 3260 3321 3325 3331 3365 3616 3669 4521
ZB8	00B8 2967 2995 3009 4528
ZB9	00B9 2814 2827 2856 2867 2911 2924 2971 3011 3013 3033 3058 3071 3079 3192 3195 3199 3284 3364 3373 3398 3402 3422 3431 4530
ZBA	00BA 2805 2845 2887 2969 3015 3077 3181 3197 3356 3371 3400 4529 4535
ZBB	00BB 3091 3104 3328 3618 3671 4522
ZBC	00BC 4523
ZBD	00BD 2356 2357 2370 2380 2395 3998 4036 4071 4104 4118 4149 4176 4223 4225 4231 4247 4254 4258 4265
ZBE	00BE 3769 4010 4018 4128 4131 4136 4243 4268 4294
ZBF	00BF 3958 3997
ZC0	00C0 1813 1817 3774
ZC1	00C1 3304 3349 3586 3600 3634 3651 4159 4445 4448 4449 4452 4453 4456
ZC2	00C2 3306 3351 3584 3603 3636 3656 4157 4443 4444 4462
ZC3	00C3 3172 3211 3279 3298 3303 4397 4402 4403
ZC4	00C4 3173 3213 3282 3301 3305 4398
ZC5	00C5 1882 1914
ZC6	00C6 1118 1120 1129 1146 1648 1910 1920 1925 2666 3506
ZC7	00C7 1244 1391 1498 1537
ZC8	00C8 1163 1176 1206 2693
ZC9	00C9 1168 1172 1522 1604 2686
ZCA	00CA 1174 2684
ZCB	00CB 1836 1864 1879 1913
ZCC	00CC 1043 1130 1789
ZCD	00CD 1042 1764 1791 1794
ZCE	00CE 1136 1801 1806
ZCF	00CF 1029 1134 1139 1796 1800
ZD0	00D0 1156 1186 1209 1334 2691
ZD1	00D1 1083 1157 1192 1370 1372 1381 1449 1460 1462 1471 1719 1740 1753 1771 1778 1798
ZD2	00D2 1081 1744 1780
ZD3	00D3 1018 1021 1064 1070 1075 1166 1175 1191 1204 1271 1273 1307 1314 1322 1335 1367 1399 1406 1422 1452 1468 1487 1491 1505 1539 1547 1562 1770 1795 2683
ZD4	00D4 1167 1200 1235 1237 1260 1385 1444 1538
ZD5	00D5 1092 1155 1272 1294 1297 1321 1378 1401 1416 1448 1457 2692
ZD6	00D6 1017 1020 1065 1069 1170 1281 1285 1286 1304 1312 1319 1403 1415 1419 1484 1486 1523 1530 1555 1570 1573 1603 1631 1649 1662 1673 2685
ZD7	00D7 1193 1195 1196 1219 1224 1328 1336 3912 3940 3957 4241 4253 4259 4260
ZD8	00D8 1247 1249 1258 1354 1474 1477 1536 1692
ZD9	00D9 1049 1058 1071 1078 1086 1287 1288 1290 1292 1298 1528 1617 1622 1629 1664 1701 1741

ZDA	00DA 1612 1619 1699 1704
ZF1	00F1 1626 1628
ZF3	00F3 975 1374 1376 1383 1464 1466 1473 1721 1773 1779 1803
ZF4	00F4 1783
ZF5	00F5 1843 1856 1880 1950
ZF6	00F6 1845 1952
ZF7	00F7 2491 2609 3146
ZF8	00F8 2898 2905 3142 3145
ZF9	00F9 2402 2553 3151
ZFA	00FA 2901 2906 3147 3150

LABEL	ESA	DECIM.	VIC	B2B4	DESCRIZIONE
D6510	0000	0	----	----	Registro di indirizzo dati 6510
R6510	0001	1	----	----	Registro ingresso/uscita 6510
	0002	2	----	----	Non usato
ADRAY1	0003-0004	3-4	0003	----	Vettore di salto.Conv. da Floating/in tero
ADRAY2	0005-0006	5-6	0005	----	Vettore di salto.Conv. da intero a Floating
CHARAC	0007	7	0007	0003	Ricerca carattere. Imm. temp. durante INT
ENDCHR	0008	8	0008	0004	Flag di ricerca apici a fine stringa
TRMPOS	0009	9	0009	----	Colonna video dopo ultimo TAB
VERCK	000A	10	000A	009D	Flag: se 0=LOAD se 1=VERIFY
COUNT	000B	11	000B	0005	Puntatore Buffer Ingresso/N. indice
DIMFLG	000C	12	000C	0006	Flag per mancanza dimens. matrici
VALTYP	000D	13	000D	0007	Flag categ. dati \$FF=stringa \$00=nume rico
INTFLG	000E	14	000E	0008	Flag categ. dati \$80=intero \$00=virgo la mobile
GARBFL	000F	15	000F	0009	Flag scans. DATA/LIST/Garbage collect tion
SUBFLG	0010	16	0010	000A	Flag indice riferimento chiamata funz z. utente
INPFLG	0011	17	0011	000B	Flag per ingresso dati: \$00=INPUT \$40 =GET \$98=READ
TANSNG	0012	18	0012	000C	Flag per segno TAN/confronto risultat i
CHANNL	0013	19	0013	000E	Flag per INPUT
LINNUM	0014-0015	20-21	0014	0011	Temp: valore intero
TEMPPT	0016	22	0016	0013	Puntatore tempo. allo Stack di string a
LASTPT	0017-0018	23-24	0017	0014	Ultimo indirizzo, temporaneo di strin ga
TEMPST	0019-0021	25-33	0019	0016	STACK tempo. per stringa
INDEX	0022-0025	34-37	0022	001F	Area di utilizzo puntatori
INDEX1	0022-0023	34-35	0022	001F	Primo Puntatore
INDEX2	0024-0025	36-37	0024	0021	Secondo Puntatore
RESHO	0026-002A	38-42	0026	0023	Risultato in virgola mobile di.multip licazione e divisione
TXTTAB	002B-002C	43-44	002B	0028	Puntatore: inizio programma Basic
VARTAB	002D-002E	45-46	002D	002A	Puntatore: inizio variabili Basic
ARYTAB	002F-0030	47-48	002D	002C	Puntatore: inizio matrici del Basic
STREND	0031-0032	49-50	0031	002E	Puntatore: Fine matrici (+1)
FRETOP	0033-0034	51-52	0033	0030	Puntatore della fine zona Immag. Stri nghe
FRESPC	0035-0036	53-54	0035	0032	Puntatore di utilita' generale per le stringhe
MEMSIZ	0037-0038	55-56	0037	0034	Puntatore indirizzo piu' alto usato

LABEL	ESA	DECIM.	VIC	B2B4	DESCRIZIONE
					dal Basic
CURLIN	0039-003A	57-58	0039	0036	Numero linea che il Basic sta trattan do
OLDLIN	003B-003C	59-60	003B	0038	Numero linea che il Basic trattava prima dell' attuale
OLDTXT	003D-003E	61-62	003D	003A	Puntatore Basic per comando CONT
DATLIN	003F-0040	63-64	003F	003C	Numero di linea in cui e' l' attuale DATA
DATPTR	0041-0042	65-66	0041	003E	Puntatore dell' indirizzo del DATA at tuale
INPPTR	0043-0044	67-68	0043	0040	Vettore per la routine di INPUT
VARNAM	0045-0046	69-70	0045	0042	Nome dell' attuale variabile
VARPNT	0047-0048	71-72	0047	0044	Puntatore dell'attuale variabile
FORPNT	0049-004A	73-74	0049	0046	Puntatore della variabile indice per FOR-NEXT
VARTXT	004B-004C	75-76	004B	0048	Immag. temp. per rout. TXTPTR durante READ,INPUT,GET
OPMASK	004d	77	004D	004A	MASK usata durante FRMEVL
TEMPF3	004E-0052	78-82	004E	004B	Immagazzinam. temp. per valore FLPT
FOUR6	0053	83	0053	0050	Lunghezza variabile stringa durante GARBAGE COLLECT
JMPER	0054-0056	84-86	0054	0051	Vett. salto usato in funz. JMP(\$4C) seguita da indir.
TEMPF1	0057-005B	87-91	0057	0054	Imm. temporaneo per valore FLPT
TEMPF2	005c-0060	92-96	005c	0059	Come sopra
FAC ----	Zona dell'accumulatore 1 per i valori in virgola mobile ----				
FACEXP	0061	97	0061	005E	FAC esponente
FACHO	0062-0065	98-101	0062	005F	FAC mantissa
FACSGN	0066	102	0066	0063	FAC segno
SGNFLG	0067	103	0067	0064	Puntatore: costante valutazione suc cessiva
BITS	0068	104	0068	0065	Indicatore di OVERFLOW per FAC 1
AFAC----	Zona dell'accumulatore 2 per i valori in virgola mobile ----				
ARGEXP	0069	105	0069	0066	AFAC esponente
ARGHO	006A-006D	106-109	006A	0067	AFAC mantissa
ARGSGN	006E	110	006E	006A	AFAC segno
ARISGN	006F	111	006F	006C	Risultato confronto segni fra FAC e AFAC
FACOV	0070	112	0070	006D	Arrotondamento del valore di FAC
FBUFPT	0071-0072	113-114	0071	006E	Puntatore Buffer di cassetta
CHRGET	0073-008A	115-138	0073	0070	Subrout.: riceve il prossimo Byte del testo Basic
CHRGOT	0079	121	0079	0076	Entrata per ottenere ancora lo stesso Byte del testo
TXTPTR	007A-007B	122-123	007A	0079	Puntatore: Byte attuale del testo Ba

LABEL	ESA	DECIM.	VIC	B2B4	DESCRIZIONE
					sic.
RNDX	008B-008F	139-143	008B	0088	Funzione RND mobile
STATUS	0090	144	0090	0096	Status W. per rout. Ker. in I/O
STKEY	0091	145	0091	009B	Flag per tasti STOP e RVS
SVXT	0092	146	0092	009C	Costante tempo. per nastro
VERCK	0093	147	0093	009D	Flags: 0=LOAD 1=VERIFY
C3PO	0094	148	0094	00A0	Flag: carattere in uscita sul Bus seriale Bufferizzato
BSOUR	0095	149	0095	00A5	Carattere memorizzato per uscita sul bus seriale
SYNO	0096	150	0096	00AB	Carattere di sincronizzazione da cassetta (vedi ST)
TEMPX	0097	151	0097	00AD	Immagazzinamento temp. di X durante l'esecuzione di CHRIN
TEMPY	0097	151	0097	00AD	Immagazz. tempo. di Y durante RS-232
LDTND	0098	152	0098	00AE	N. files aperti/ indice tavola Files
DFLNT	0099	153	0099	00AF	Standard ingresso dati (0=Tastiera)
DFLTO	009A	154	009A	00B0	Standard uscita dati (CMD) (3=video)
PRTY	009B	155	009B	00B1	Carattere di parita' del nastro
DPSW	009C	156	009C	00B2	Flag: ricevuto Byte da nastro
MSGFLG	009D	157	009D	----	Flag: \$00=m. programma, soppressione messaggi di errore. \$40=m. Kernal invio errori. \$80=m.diretto, tutti gli errori
PTR1	009E	158	009E	00C0	Errore logico da nastro. Passo 1
PTR2	009F	159	009F	00C1	Errore logico da nastro. Passo 2
TIME	00A0-00A2	160-162	00A0	00B0	Orologio Int.(CLOCK) in tempo reale
TSFCNT	00A3	163	00A3	00B7	Contat. per Bit da nastro R.o W./FLG
TBTCNT	00A4	164	00A4	00B9	Conteggio ciclo
BUFPT	00A6	166	00A6	00BB	Puntatore: Buffer di I/O nastro
INBIT	00A7	167	00A7	----	RS232 imm. temp. ricez. bit/Temporizz. di cassetta
BITCI	00A8	168	00A8	00BE	RS232 contatore ingresso bit/tempo.
di cassetta					
RINONE	00A9	169	00A9	00BF	Flag per RS232: controllo bit di partenza
RIDATA	00AA	170	00AA	00C2	Buffer ingresso byte per RS232/Temporizzazione cassetta
RIPRTY	00AB	171	00AB	00C3	INPUT di parita' per RS232/Controllo cassetta
SAL	00AC-00AD	172-173	00AC	00C7	Puntatore: Buffer del nastro/scrolling di schermo
EAL	00AE-00AF	174-175	00AE	00C9	Indirizzo di fine nastro/ Fine del programma
TAPE1	00B2-00B3	178-179	00B2	00D6	Puntatore inizio Buffer nastro
BITTS	00B4	180	00B4	00CE	Cont. bit uscita per RS232/Flag tempo orizzazione lettura nastro

LABEL	ESA	DECIM.	VIC	B2B4	DESCRIZIONE
NXTBIT	00B5	181	00B5	00CF	Prossimo bit da inviare a RS232/Lett.
nastro per EOT					
RODATA	00B6	182	00B6	00D0	Buffer per Byte in uscita su RS232/Fl ag errore di lettura nastro
FNLEN	00B7	183	00B7	00D1	N. di caratteri del nome del File
LA	00B8	184	00B8	00B2	N. dell'attuale file logico
SA	00B9	185	00B9	00D3	Attuale indirizzo secondario
FA	00BA	186	00BA	00D4	Attuale n. di periferica
FNADR	00BB-00BC	187-188	00BB	00D5	Puntatore: attuale indirizzo del nome del file
ROPRTY	00BD	189	00BD	00DD	Parita' di uscita per RS232/Temp. di cassetta
FSBLK	00BE	190	00BE	00DE	Contatore dei blocchi in I/O per nast ro
MYCH	00BF	191	00BF	00DF	Buffer per porta seriale
CAS1	00C0	192	00C0	00F9	Interruttore per motore cassetta
STAL	00C1-00C2	193-194	00C1	00FB	Indirizzo di partenza per LOAD e per scrittura cassetta
MEMUSS	00C3-00C4	195-196	00C3	----	Temp. di LOAD per cassetta
LSTX	00C5	197	00C5	0097	Valore ultimo tasto premuto. \$40=ness un tasto (64/VIC), \$FF=(PET)
NDX	00C6	198	00C6	009E	N. di caratteri nella coda di tastier a (BUFFER-QUEUE)
RVS	00C7	199	00C7	009F	Flag per REVERSE \$01=ON \$00=OFF
INDX	00C8	200	00C8	00A1	Puntatore fine linea per INPUT
LXSP	00C9-00CA	201-202	00C9	00A3	Posizioni X-Y cursore a inizio Input
SFDX	00CB	203	----	----	Flag: stampa caratteri shiftati
KEYVAL	----		00CB	00A6	Valore della matrice del tasto premut o durante l' ultima scansione di tast iera
BLNSW	00CC	204	00CC	00A7	Lampeggio cursore \$00=Abilitato \$01=d isabilitato
BLNCT	00CD	205	00CD	00A8	Temporizzatore di conto alla rovescia per lampeggio cursore
GDBLN	00CE	206	00CE	00A9	Carattere presente sotto il cursore
BLNON	00CF	207	00CF	00AA	Flag di stato del cursore \$00=OFF \$01 =ON
CRSW	00D0	208	00D0	00AC	Flag: Input da schermo=\$03, da tastie ra = \$00
PNT	00D1-00D2	209-210	00D1	00C4	Puntatore all' attuale linea di indir izzo schermo
PNTR	00D3	211	00D3	00C6	Posizione del cursore sulla colonna d ell' attuale linea
QTSW	00D4	212	00D4	00CD	Indicatore di cursore fra apici.\$00=n o
LNMX	00D5	213	00D5	00D5	Lunghezza dell' attuale linea di sche rmo

LABEL	ESA	DECIM.	VIC	B2B4	DESCRIZIONE
TBLX	00D6	214	00D6	00D8	Attuale linea di schermo del cursore
SCHAR	00D7	215	00D7	00D9	Valore dell' attuale carattere in ingresso/ Ultimo carattere uscito
INSRT	00D8	216	00D8	00DC	Flag: n. di volte dell' attuale tasto INST
LDTB1	00D9-00F2	217-242	00D9	00E0	Tavola di collegamento linee schermo/ Immagazzinamento temp. EDITOR
USER	00F3-00F4	243-244	00F3	----	Puntatore per attuale colore di schermo in RAM
KEYTAB	00F5-00F6	245-246	00F5	----	Vettore alla tavola di decodifica tastiera
RIBUF	00F7-00F8	247-248	00F7	----	Puntatore Buffer ingresso RS232
ROBUF	00F9-00FA	249-250	00F9	----	Puntatore Buffer uscita RS232
FREKZP	00FB-00FE	251-254	00FB	----	Spazio libero per programmi utente in pagina ZERO
BASZPT	00FF	255	00FF	----	Area per dati Basic
ASCWRK	00FF-010A	255-266	00FF	00FF	Area assembler per conversione virgola mobile in ASCII
BAD	0100-013E	256-***	0100	0100	Errore di input da nastro
STACK	0100-01FF	256-***	0100	0100	Area di STACK hardware per 6510
BSTACK	013F-01FF	319-***	013F	013F	Area di STACK per il Basic
BUF	0200-0258	512-600	0200	----	Buffer di INPUT di sistema/ Da 0200 a 0250 per PET
LAT	0259-0262	601-610	0259	0251	Tavola KERNAL: numero del file logico attivo
FAT	0263-026C	611-620	0263	025B	Tavola KERNAL: numero di periferica per ogni file
SAT	026D-0276	621-630	026D	0265	Tavola KERNAL: indirizzo secondario di ogni file
KEYD	0277-0280	631-640	0277	026F	Coda del BUFFER di tastiera. (metodo FIFO)
MEMSTR	0281-0282	641-642	0281	----	Puntatore al punto piu' basso della memoria per Sistema Operativo
MEMSIZ	0283-0284	643-644	0283	----	Puntatore al massimo della memoria per il Sistema Operativo
TIMOUT	0285	645	0285	----	Flag: variabile KERNAL per fuori temporizzazione IEEE
COLOR	0286	646	0286	----	Codice dell' attuale colore del carattere
GDCOL	0287	647	0287	----	Colore di fondo sotto il cursore
HIBASE	0288	648	0288	----	Byte alto indirizzo memoria schermo
XMAX	0289	649	0289	----	Massimo numero di Bytes nel Buffer di tastiera
RPTFLG	028A	650	028A	----	Flag di REPEAT \$00=Cursore, INST, spazio/\$40=Nessun tasto/\$80= Tutti i tasti
KOUNT	028B	651	028B	----	Contatore di velocita' di REPEAT

LABEL	ESA	DECIM.	VIC	B2B4	DESCRIZIONE
DELAY	028C	652	028C	----	Contatore del ritardo di REPEAT
SHFLAG	028D	653	028D	0098	Flag tasto SHIFT.\$00=Nessun tasto \$01=Shift, \$02=CBM, \$04=CTRL (i valori si sommano es. \$07=SHIFT+CBM+CTRL)
LSTSHF	028E	654	028E	----	Ultimo tasto shift premuto
KEYLOG	028F-0290	655-656	028F	----	Vettore: Routine per determinare la tavola di tastiera
MODE	0291	657	0291	----	Flag: cambiamento set-\$00=disabilitato, \$80=abilitato
AUTODN	0292	658	0292	----	Flag: scorrimento video-\$00=disabilitato
M51CTR	0293	659	0293	----	RS232/registro di controllo immagini del 6551
M51CDR	0294	660	0294	----	RS232/registro di comando immagine del 6551
M51AJB	0295-0296	661-662	0295	----	RS232/non-standard, non utilizzato nel VIC in EUROPA
RSSTAT	0297	663	0297	----	RS232/registro di immagine del 6551
BITNUM	0298	664	0298	----	Numero di bits rimasti da inviare
BAUDOF	0299-029A	665-666	0299	----	RS232 baud rate-bit in microsecondi
RIDBE	029B	667	029B	----	Indicatore RS232 al termine del buffer di INPUT
RIDBS	029C	668	029C	----	Puntatore di RS232: Byte alto dell'indirizzo del buffer di ingresso dati
RODBS	029D	669	029D	----	Puntatore RS232: Byte alto dell'indirizzo del buffer di uscita
RODBE	029E	670	029E	----	Indice RS232 al termine del buffer di uscita
IRQTMP	029F-02A0	671-672	029F	----	Immagazzinamento temporaneo del vettore IRQ durante le operazioni su nastro
ENABL	02A1	673	02A1	----	Abilitazione dell' RS232
TODSNS	02A2	674	----	----	Controllo sensore cassetta durante le operazioni di I/O
TRDTMP	02A3	675	----	----	Immagazzinamento temporaneo durante la lettura nastro
TD1IRQ	02A4	676	----	----	Indicatore temporaneo di IRQ durante la lettura del nastro
TLNIDX	02A5	677	----	----	Indicatore temporaneo per indice linea di schermo
TVSFLG	02A6	678	----	----	Flag per TV-\$00=NTSC, \$01=PAL
	02A7-02FF	679-767	02A1	----	Area attualmente non utilizzata
IERROR	0300-0301	768-769	0300	----	Vettore: ingresso indiretto al messaggio di errore Basic,(X) indica il messaggio. Il contenuto normale e', nel CBM64 \$E38B

LABEL	ESA	DECIM.	VIC	B2B4	DESCRIZIONE
IMAIN	0302-0303	770-771	0302	----	Vettore: ingresso indiretto per linea Basic e decodifica. CBM64=\$A483
ICRNCH	0304-0305	772-773	0304	----	Vettore: ingresso indiretto alla routine Basic codificata (TOKEN). CBM64=\$A57C
IQPLOP	0306-0307	774-775	0306	----	Vettore: ingresso indiretto alla routine Basic LIST. CBM64=\$A71A
IGONE	0308-0309	776-777	0308	----	Vettore: ingresso indiretto per la ricerca in Basic delle parole chiave. CBM64=\$A7E4
IEVAL	030A-030B	778-779	030A	----	Vettore: ingresso indiretto al Basic per l'elaborazione dei comandi (TOKEN). CBM64=\$AEB6
SAREG	030C	780	030C	----	Immagazzinamento di A durante l'esecuzione di SYS
SXREG	030D	781	030D	----	Immagazzinamento di X durante l'esecuzione di SYS
SYREG	030E	782	030E	----	Immagazzinamento di Y durante l'esecuzione di SYS
SPREG	030F	783	030F	----	Immagazzinamento di SP durante l'esecuzione di SYS
USRPOK	0310	784	0000	0000	Funzione USR per JMP
USRADD	0311-0312	785-786	0001	0001	Indirizzo USR Byte basso/Byte alto
	0313	787			Non usato
CINV	0314-0315	788-789	0314	0090	Vettore: indirizzo hardware per l'interrupt di IRQ. CBM64=\$EA31
CNBINV	0316-0317	790-791	0316	0092	Vettore: indirizzo di interrupt per l'istruzione BRK. CBM64=\$FE66
NMINV	0318-0319	792-793	0318	0094	Vettore: Indirizzo Hardware di interrupt per l'istruzione NMI. CBM64=\$FE47
IOPEN	031A-031B	794-795	031A	----	Vettore: ingresso indiretto per la routine Kernal OPEN. CBM64=\$F34A
ICLOSE	031C-031D	796-797	031C	----	Vettore: ingresso indiretto per la routine Kernal CLOSE. CBM64=\$F291
ICKIN	031E-031F	798-799	031E	----	Vettore: ingresso indiretto per la routine Kernal CHKIN. CBM64=\$F20E
ICKOUT	0320-0321	800-801	0320	----	Vettore: ingresso indiretto per la routine Kernal CHKOUT. CBM64=\$F250
ICLRCH	0322-0323	802-803	0322	----	Vettore: ingresso indiretto per la routine Kernal CLRCHN. CBM64=\$F333
IBASIN	0324-0325	804-805	0324	----	Vettore: ingresso indiretto per la routine Kernal CHRIN. CBM64=\$F157
IBSOUT	0326-0327	806-807	0326	----	Vettore: ingresso indiretto per la routine Kernal CHROUT. CBM64=\$F1CA
ISTOP	0328-0329	808-809	0328	----	Vettore: ingresso indiretto per la routine Kernal CHROUT. CBM64=\$F1CA

LABEL	ESA	DECIM.	VIC	B2B4	DESCRIZIONE
IGETIN	032A-032B	810-811	032A	----	utine Kernal STOP. CBM64=\$F6ED Vettore: ingresso indiretto per la ro utine Kernal GETIN.CBM64=\$F13E
ICLALL	032C-032D	812-813	032C	----	Vettore: ingresso indiretto per la ro utine Kernal CLALL.CBM64=\$F32F
USRCMD	032E-032F	814-815	03FA	----	Vettore definito dall' utente. Il con tenuto normale e' \$FE66
ILOAD	0330-0331	816-817	0330	----	Vettore: ingresso indiretto per la ro utine Kernal LOAD. CBM64=\$F4A5
ISAVE	0332-0333	818-819	0332	----	Vettore: ingresso indiretto per la ro utine Kernal SAVE. CBM64=\$F5ED
	0334-033B	820-827	0334	----	Normalmente non utilizzato
TBUFFR	033C-03FB	828-1019	033C	027A	Buffer del nastro per operazioni di I /O
	03FC-03FF	1020-1023	03FC	----	Normalmente non utilizzato
VICSCN	0400-07FF	1024-2047	1E00	8000	Area di memoria dello schermo

LABEL C64 V20 B2.0 B4.0 DESCRIZIONE

BCOLD	A000	C000	----	----	Vettore Basic di RESET iniziale
BWARM	A002	C002	----	----	Vettore Basic di RESET con INTERRUPT
STNDSP	A00C	C00C	C000	B000	Tavola dei vettori dei comandi Basic
FUNDSP	A052	C052	C046	B066	Tavola dei vettori delle funzioni Basic
OPTAB	A080	C080	C074	B094	Vettore degli operatori Basic e tavola di priorit� - 2 bytes di indirizzo e 1 di priorit�
RESLST	A09E	C09E	C092	B0B2	Tavola delle parole chiave
MSCLST	A129	C129	C11D	B13D	Tavola mista delle parole chiave
OPLIST	A140	C140	C134	B161	Tavola degli operatori delle parole chiave
FUNLST	A14D	C14D	C141	B16E	Tavola delle funzioni
ERRTAB	A19E	C19E	C192	B20D	Messaggi di errore
ERRPTR	A328	C328	----	----	Puntatori ai messaggi di errore
OKK	A364	C364	----	----	Nessun messaggio di errore-"ok","error","in","ready","break".
FNDFOR	A38A	C38A	C2AA	B322	Cerca il punto di entrata di FOR nello Stack o saltalo per cercare l'entrata di GOSUB dopo un RETURN
BLTU	A3B8	C3B8	C2D8	B350	Sposta in alto un blocco di memoria-controllo per memoria sufficiente
BLTUC	A3BF	C3BF	C2DF	B357	Sposta un blocco da (LOWTR) a (HIGHTR)
GETSTK	A3FB	C3FB	C31B	B393	Controllo dello Stack per l'immissione di due bytes dell'accumulatore-errore di "OUT OF MEMORY" se non c'� posto
REASON	A408	C408	C328	B3A0	Controllo che l'indirizzo (A/Y) sia minore dello spazio per le stringhe- se no...
OMERR	A435	C435	C355	B3CD	Stampa un messaggio di "OUT OF MEMORY"
ERROR	A437	C437	C357	B3CF	Stampa il messaggio di errore indicato da (X) poi ...
ERRFIN	A469	C469	C37A	B3FO	Stampa "ERROR" o "BREAK"
READY	A474	C474	C389	B3FF	Restart del BASIC - stampa "READY" poi ...
MAIN	A480	C480	C392	B406	Linea di INPUT - identifica un comando o una linea basic
MAINI	A4A2	C4A2	C3AB	B41F	Se � una linea basic allora immetti il numero di linea e converti le parole chiave in TOKENS
INSLIN	A4A2	C4A2	C3B1	B47Q	Inserisci un testo dal buffer del Basic entro un programma - il numero di linea in (LINNUM) in entrata - La linea deve avere le parole chiave cambiate in tokens e la lunghezza della linea in (Y)- se (BBUFF)=\$00 allora la linea sar� cancellata - la routine esiste come conseguenza di MAIN
FINI	A52A	C52A	C439	B4AD	Dopo l'inserimento di una nuova linea nel test o basic esegui le routine di RUNC, LNKPRG e rientra alla routine MAIN
LNKPRG	A533	C533	C442	B4B6	Ricostruisce i puntatori di LINK del testo BASIC
INLIN	A560	C560	C46F	B4E2	Inserisce una linea nel buffer del basic-immetti

LABEL C64 V20 B2.0 B4.0 DESCRIZIONE

					ti OO al termine
RDCHR	----	----	C481	----	Immetti un carattere nell' accumulatore -se il carattere e' OF cioe' LINE-FEED inverti il flag in OD per cancellare l'uscita
CRUNCH	A579	C579	C495	B4FB	Cambia le parole chiave in Tokens - la linea in n BBUFF -fissa TXTPTR a BBUFF- Y alla lunghezza della linea e TXTPTR a BBUFF-1 in uscita
FNDLIN	A613	C613	C52C	B5A3	Ricerca il testo BASIC dall' inizio per il numero della linea in (LINNUM)...oppure...
FNDLNC	A617	C617	C530	B5A7	Ricerca il testo BASIC da (A/X) per il numero di linea in (LINNUM)- se lo trova fissa C e (LINPTR) all' inizio della linea- azzera C
SCRATH	A642	C642	C55B	B5D2	Punto di entrata di NEW- controllo di sintassi poi.....
SCRATCH	A659	C659	C55D	B5D4	Metti il primo Byte di testo a OO-fissa(VARTAB) a (TXTTAB)+2 poi...
RUNC	A659	C659	C572	B5E9	Resetta l' esecuzione all' inizio del programma (STXPTR) e dopo esegui la routine CLEARC
CLEAR	A65E	C65E	C577	B5EE	Punto di entrata per CLEAR-controllo di sintassi poi....
CLEARC	A660	C660	C579	B5FC	Fissa (FRETOP) a (MEMSIZ)-chiude le funzioni di I/O-fissa (ARYTAB) a (VARTAB) poi...
LDCLR	A677	C677	C590	B60B	Esegui RESTOR-azzera (TEMPPT)-azzera lo STACK
STXPT	A68E	C68E	C5A7	B622	Metti (TXTPTR) al valore di (TXTTAB)-1 per resettare l' esecuzione all' inizio programma.
LIST	A69C	C69C	C5B5	B630	Punto di ingresso alla routine LIST
QPLOP	A717	C717	C63A	B6B5	Manipolazione dei caratteri in conseguenza del comando LIST-se trattasi di NON-TOKEN o TOKEN fra virgolette, allora stampali come sono. In caso contrario espandi i TOKENS e stampali
FOR	A742	C742	C65B	B6DE	Punto di ingresso alla routine FOR-salva(TXTPTR), (CURLIN) ed il valore finale nello STACK, dopo....
NEWSTT	A7AE	C7AE	C6C4	B74A	Controllo per tasto di RUN/STOP premuto-
CKEOL	A7C4	C7C4	C6DA	B7F5	Controlla che la fine della linea sia anche la fine del testo - in caso contrario vai ai successivi parametri nella stessa linea
GONE	A7E1	C7E1	C6F7	B77C	Esegue il comando entro la linea
GONE3	A7ED	C7ED	C700	B785	Interpreta un comando basic e lo esegue
RESTOR	A81D	C81D	C730	B7B7	Punto di entrata per il comando RESTORE - azzera (DATPTR) all'inizio del basic
STOP	A82C	C82C	C73F	B7C6	Punto di ingresso per comando STOP - azzera il Carry per il messaggio di "Break" poi salta entro la routine di END
END	A82F	C82F	C741	B7C8	Punto di ingresso per il comando END - Fissa il Carry poi

LABEL	C64	V20	B2.0	B4.0	DESCRIZIONE
FINID	A834	C834	C744	B7CB	Se indiretto salva (TXTPTR) in (OLDTXT) poi..
STPEND	A841	C841	C751	B7DB	Salva (CURLIN) in (OLDLIN) e vai alla routine di READY se il Carry = END) se il Carry e' a zero vai ERRFIN
CONT	A857	C857	C76B	B7EE	Punto di ingresso per il comando CONT - riposizione (TXTPTR) e (CURLINE) fino a quando (OLDTXT) non sia 0 (CAN'T CONTINUE)
RUN	A871	C871	C785	B808	Punto di ingresso per RUN - esegue CLR e poi GOTO
GOSUB	A883	C883	C790	B813	Punto di ingresso per GOSUB - salva (TXTPTR), (CURLIN) e il Flag di GOSUB (8D) nello Stack e poi esegue il GOTO
GOTO	A8A0	C8A0	C7AD	B830	Punto di ingresso per GOTO - legge il numero d al testo basic e lo inserisce in (LINNUM) poi.
GOTOC	A8A3	C8A3	C7B0	B833	Esegue la ricerca per la fine della linea attuale - ricerca la linea (LINNUM) e fissa (TXTPTR) quando lo trova
RETURN	A8D2	C8D2	C7DA	B85D	Punto di ingresso per RETURN - controlla la sintassi poi...
RTC	A8D4	C8D4	C7DC	B85F	Esegue l'azzeramento dello Stack fino al primo punto di entrata di gosub - fissa poi (TXTPTR) e (CURLIN) dallo Stack
DATA	A8F8	C8F8	C800	B883	Punto di ingresso per DATA - esegue la ricerca del testo per trovare la fine del comando
DATAN	A906	C906	C80E	B891	Fissa la ricerca per i delimitatori del comando (Virgola o Byte 0) dopo ricerca ...
SERCHX	A90B	C90B	C811	B894	Ricerca nel testo (X) o Byte 0 - esce con (Y) fissato al numero di Byte da delimitare.
IF	A92B	C92B	C830	B8B3	Punto di ingresso per IF - valuta l' espressione esegue la routine REM se IF= (falso)
REM	A93B	C93B	C843	B8C6	Punto di ingresso per REM - esegue la ricerca per il Byte 0
DOCOND	A940	C940	C848	B8CB	Se la condizione relativa a IF e' non 0 (cioe' vera) allora esegui il comando o GOTO
ONGOTO	A94B	C94B	C853	B8D6	Punto di ingresso per ON - prende un numero dal testo e ricerca il numero di linea esegui GOTO o GOSUB
LINGET	A96B	C96B	C873	B8F6	Legge un intero dal testo e lo immette in (LINNUM) - da un errore se il valore non fra 0 e 63999
LET	A9A5	C9A5	C8AD	B930	Punto di ingresso per LET - cerca la variabile obbiettivo nello spazio delle variabili e fissa (FORPNT) con l'indirizzo della variabile - e labora l'espressione ed esegue PUTINT o PTFLPT o PUTTIM o GETSPT
PTFLPT	A9C4	C9C4	C8CC	B94F	Immette l'accumulatore in virgola mobile (FAC)

					entro la variabile il cui indirizzo e' dato da (FORPNT).
PUTINT	A9D6	C9D6	C8DE	B961	Immette il valore intero contenuto in (FAC+3) entro la variabile il cui indirizzo e' in (FORPNT)
PUTTIM	A9E3	C9E3	C8EB	B972	Fissa TI\$-fissa(INDEX1) in modo taleche punti alla stringa e (A) a 6 che e' la lunghezza della stringa
GETSPT	AA2C	CA2C	C937	B9BA	Immetti l' identificatore della stringa punato da (FAC+3) entro la variabile stringa a cui punta (FORPNT).
PRINTN	AA80	CA80	C98B	BA88	Punto di ingresso per PRINTf - esegue CMD per abilitare il Device corrente di I/O (manda l' Unlisten alla IEEE se il numero di device e' 3)
CMD	AA86	CA86	C991	BA8E	Punto di ingresso per CMD - abilita al CMD il device definito dalla tavola di PRINT
STRDON	AA9A	CA9A	C9A5	BAA2	Routine di PRINT - stampa la stringa e controlla per la fine della linea di PRINT
PRINT	AAA0	CAA0	C9AB	BAA8	Punto di ingresso per PRINT - identifica i parametri di PRINT (SPC, TAB etc) - valuta l' espressione.
VAROP	AAB8	CAB8	C9C3	BAC0	Uscita variabile-(se numero lo converte in stringa)-uscita stringa
NUNDON	AABC	CABC	C9C7	BAC4	Stampa routine-stampa numerico
CRDO	AAD7	CAD7	C9E2	BADF	Uscita CR/LF-nel VIC e CBM64 se (CHANNL) maggiore di 127 solo uscita CR
COMPTR	AAE8	CAE8	C9EF	BAF0	Stampa tabulazioni e spazi per delimitatori virgola-funzioni SPC o TAB
STROUT	AB1E	CB1F	CA1C	BB1D	Stampa la stringa puntata da (A/Y) fino a che non trova 1 byte a zero
STRPTR	AB21	CB21	CA1F	BB20	Stampa la stringa puntata da (FAC 3) finche' non trova un Byte a zero
OUTSTR	AB24	CB24	CA22	BB23	Stampa la stringa puntata da (INDEX1) di lunghezza (A)
OUTSPC	AB3B	CB3B	CA39	BB3A	Stampa spazio
PRTSPC	AB3F	CB3F	CA3D	BB3E	Stampa SEMPRE uno spazio
OUTSKP	AB42	CB42	CA40	BB41	Stampa sempre il cursore a destra
OUTQST	AB45	CB45	CA43	BB44	Stampa punto interrogativo
OUTDO	AB47	CB47	CA45	BB46	Stampa (A)
TRMNOK	AB4D	CB4D	CA4F	BB4C	Manipolazione dei messaggi di errore per GET, INPUT e READ
GET	AB7B	CB7B	CA7D	BB7A	Punto di ingresso per GET- Controlla che non sia un comando diretto, identifica GETf, immette un carattere
INPUTN	ABA5	CBA5	CAA7	BBA4	Punto di ingresso per INPUTf-fissa la periferica di input, esegue l' input poi mette la IEEE

INPUT	ABBF	CB8F	CAC1	BBBE	in UNLISTEN se la periferica e' maggiore di 3 Punto di ingresso per comando INPUT- visualizza un messaggio se esiste, esegue quindi l' input
BUFFUL	ABEA	CBAE	CAED	BBE8	Legge l' input-se (BBUFF) e' zero, cioe' nessuna stringa in input il VIC 20 ed il CBM&\$ saltano, BASIC 2/4 arrestano l' operazione
QINLIN	ABF9	CBF9	CAFA	BBF5	Stampa ? e immette i dati nel buffer (BBUFF)
READ	AC06	CC06	CB07	BC02	Punto di ingresso per il comando READ-fissa il flag di READ(98) in (INPFLG), fissa (X/Y)=(DATPTR)
INPCON	AC0D	CC0D	CB0E	BC09	Punto di ingresso a READ per INPUT-fissa il flag di INPUT(00) in (INPFLG), fissa (X/Y)=BUF
INPC01	ACOF	CCOF	CB10	BC0B	Punto di ingresso di READ per GET-fissa il
RDGET	AC35	CC35	CB36	BC31	Parte della routine READ che esegue il GET di 1 Byte
RDINP	AC43	CC43	CB44	BC3F	Parte della routine di READ che esegue il comando INPUT usando la routine RDGET
DATLOP	ACB8	CCB8	CBB9	BCB4	Parte della routine di READ che esegue il READ dei DATA, usando RDGET
EXINT	ACFC	CCFC	CBFC	BCF7	Stringa ASCII-"?EXTRA IGNORED"
TRYAGN	ADOC	CDOC	CCOD	RD0T	Stringa ASCII-"?REDO FROM START"
NEXT	AD1E	CD1E	CC20	BD19	Punto di ingresso per NEXT-immette la successiva (NEXT) variabile e conferma che il corrispondente comando FOR sia nello STACK, calcola il prossimo valore di variabile ciclo
DONEXT	AD61	CD61	CC62	BD58	Se il contatore di ciclo e' un valore valido, fissa(CURLIN) e (TXTPTR) dallo STACK e reimmette il FOR nel ciclo
CHKNUM	AD8D	CD8D	CC8E	BD87	Controlla la routine (VALTYP) per i risultati numerici provenienti da (FRMEVL-vedi dopo)-va in READY con un "TYPE MISMATCH" se e' rilevato un risultato numerico
CHKNUM	AD8F	CD8F	CC90	BD89	Controlla la routine (VALTYP) per risultato stringa-il resto vedi precedente
FRMNUM	AD8A	CD8A	CC8B	BD84	Valuta una espressione numerica dal testo Basic, mette in funzione la routine FRMELV(vedi dopo) poi la (CHKNUM)
FRMEVL	AD9E	CD9E	CC9E	BD98	Immette e valuta una qualsiasi espressione del testo Basic. Fissa VALTYP (00 se numerica, FF se stringa) e INTFLG (00 se in virgola mobile, 80 se intero) Se l' espressione e' numerica in virgola mobile, il risultato e' immesso in FAC. Se l' espressione e' un intero numerico, allora il risultato e' riportato in FAC+3 nel formato HI/LO. Se si tratta di un' espressione stringa, allora i

					1 puntatore alla descrizione della stringa e' riportato in (FAC+3). Questo e' di solito un copia di VARPNT. In aggiunta se l' espressione e' una variabile semplice, allora la routine VARNAM puntera' al primo Byte del nome. Per concludere, se sara' rilevato un errore nell' espressione allora torneremo in Basic con READY preceduto da "SYNTAX ERROR"
EVAL	AE83	CE83	CD84	BE81	Valuta il singolo termine in un' espressione. Identifica le funzioni pi greco, TI, TI\$, ecc.
PIVAL	AEA8	CEA8	CDA3	BEA0	Valore in virgola mobile di pi-greco nel formato MFLPT.-3.1415965
QDOT	AEAD	CEAD	CDAB	BEA5	Valuta un termine non variabile in un' espressione
PARCHK	AEF1	CEF1	CDEC	BEE9	Valuta un' espressione in parentesi
CHKCLS	AEF7	CEF7	CDF2	BEEF	Controlla che il carattere puntato(indirizzato) da TXTPTR sia una parentesi destra. Se non e' cosi' stampa "SYNTAX ERROR"
CHKOPN	AEFA	CEFA	CDF5	BEF2	Controllo come sopra per parentesi sinistra
CHKCOM	AEFD	CEFD	CDF8	BEF5	Controlla che il carattere a cui punta TXTPTR sia una virgola. Se non e' visualizza "SYNTAX ERROR"
SYNCHR	AEFF	CEFF	CDF8	BEF7	Controlla che il carattere puntato da TXTPTR sia lo stesso contenuto in Accumulatore, altrimenti vedi sopra
SYNERR	AF08	CF08	CE03	BF00	Stampa il messaggio di errore "SYNTAX ERROR" e ritorna in modo Basic con lla scritta Ready
RSVVAR	AF14	CF14	----	----	Setta il Carry se la variabile puntata da (FAC+3) e' una parola riservata es. ST, TI, TI\$
ISVAR	AF28	CF28	CE0F	BF04	Cerca una variabile con nome nel testo Basic. Fissa (VARNAM) a puntare al nome della variabile nella tavola se e' trovato. Immette un valore numerico in FAC, il puntatore stringa in (FAC+3).
TISASC	AF48	CF48	CE2E	BFAD	Converte TI in una stringa ASCII e fissa FAC+1 a puntare alla stringa
ISFUN	AF47	CF47	CE89	C037	Valuta la funzione. Riporta il valore numerico in FAC, il valore stringa come puntatore in FAC+3
STRFUN	AFB1	CFB1	CE93	C051	Salva l' indicatore stringa di una funzione stringa nello Stack e lo valuta
NUMFUN	AFD1	CFD1	CEB3	C071	Valuta l' argomento di una funzione numerica e calcola il valore della funzione
OROP	AFE6	CFE6	CECB	C086	Esegue OR. Fissa il flag di OR e usa ANDOP per la valutazione della funzione
ANDOP	AFE9	CFE9	CECB	C089	Esegue AND. Fissa il flag di AND poi converte il valore in virgola mobile a fisso, esegue AN

					D (o OR se e' fissato il flag dii OR) poi ric onverte in virgola mobile
DOREL	B016	D016	CEF8	COB6	Esegue una delle relazioni (maggiore, minore o uguale). Se e' un'espressione numerica usa NUM REL, se e' un' espressione stringa usa STRREL
NUMREL	B018	D018	CEFD	COB8	Esegue un confronto numerico
STRREL	B02E	D02E	CF10	COCE	Esegue un confronto stringa
DIM	B081	D081	CF63	C121	Esegue DIM
PPTRGET	B08B	D08B	CF6D	C12B	Identifica una variabile con nome nel testo Ba sic e piazzail nome, non il puntatore al nome, in (VARNAM)
ORDVAR	B0E7	DOE7	CFC9	C187	Cerca una variabile il cui nome sia in VARNAM e fissa (VARPNT) perche' ci punti. Se necessar io usa NOTFNS per creare una nuova variabile
ISLETC	B113	D113	CF77	C1B6	Setta il Carry se nell' accumulatore e' presen te una lettera
NOTFNS	B11D	D11D	D001	C1C0	Crea una nuova variabile con nome come in (VAR NAM), a meno che PTRGET sia chiamata in funzio ne da ISVAR
NOTEVL	B128	D128	D00C	C1CB	Crea una nuova variabile con nome come in (VAR NAM) e fissa (VARPNT) per puntare ad essa
FMAPTR	B194	D194	D078	C2C8	Fissa (ARYPNT) all' inizio della matrice ed im mette il numero di dimensioni della matrice in COUNT
N32768	B1A5	D1A5	D089	C2D9	Valore in virgola mobile di 32768 in formato F LPT
FACINX	B1AA	D1AA	-----	-----	Converte (FAC) in intero in (A/Y)
INTIDX	B1B2	D1B2	D08D	C2DD	Valuta un' espressione presente nel testo Basi c come intero nell' intervallo -32768 e +32767
AYINT	B1BF	D1BF	D09A	C2EA	Valuta l' espressione in testo Basic come inte ro positivo (da 0 a +32767)
ISARY	B1D1	D1D1	D0AC	C2FC	Prende i parametri delle matrici dal testo Bas ic e li immette nello Stack
FNDARY	B218	D218	D0F3	C343	Trova la matrice-il nome in VARNAM, i parametr i sono letti da ISARY
NOTFDD	B261	D261	D13C	C38C	Crea una matrice dai parmerti nello Stack
INLPN2	B30E	D30E	D1EA	C439	Fissa (VARPNT) per pntare ad un' elemento entr o la matrice
UMULT	B34C	D34C	D228	C447	Calcola il numero di Bytes nella dimensione (Y) della matrice con inizio a (VARPNT)
FRE	B37D	D37D	D259	C4A8	Punto di ingresso per la funzione FRE-esegue l a garbage collection e fissa il valore della f unzione a (FRETOP)-(STREND)
GIVAYF	B391	D391	D26D	C4BC	Converte l'intero in (A/Y) in FLPT in FAC nell intervallo fra 0 32767
SNGFT	B3A2	D3A2	D27C	C4CB	Converte (Y) a formato FLPT in (FAC) nell' int

LABEL C64 V20 B2.0 B4.0 DESCRIZIONE

					ervallo da 0 a 255
ERRDIR	B3A6	D3A6	D280	C4CF	Stampa "ILLEGAL DIRECT", se e' in modo diretto - per esempio (CURLIN) =\$FF
DEF	B3B3	D3B3	D28D	C4DC	Punto di ingresso per DEF-crea la funzione FN
GETFNM	B3E1	D3E1	D2BB	C50A	Controlla la sintassi di FN, localizza l' indicatore di FN e fissa DEFINT perche' punti all' indicatore
FNDEOR	B3F4	D3F4	D2CE	C51D	Punto di ingresso per la funzione FN-riceve l' indicatore di FN, poi...
SETFNV	B423	D423	D2FD	C54C	Fissa (TXTPTR) all' inizio di FN nel testo, valuta l' espressione,resetta quindi (TXTPTR)
STRD	B465	D465	D33F	C58E	Punto di ingresso per la funzione STR\$, valuta l' espressione e la converte in stringa ASCII
STRINI	B475	D475	D34F	C59E	Crea spazio per la stringa il cui indicatore e ' in FAC 3 e la cui lunghezza in (A). Termina con un nuovo indicatore in (DSCTMP) ed un puntatore al vecchio indicatore e' in (DSCPNT)
STRILT	B487	D487	D361	C5B0	Esamina la stringa che parte da (A/Y) e crea un indicatore. Termina con (FAC 3) che punta al l' indicatore
PUTNW1	B4D5	D4D5	D3AF	----	Fissa l' indicatore allo Stack indicatore e ri crea il puntatore
GETSPA	B4F4	D4F4	D3CE	C61D	Fissa (FRETOP) e (FRESPC) per una nuova stringa la cui lunghezza e' in (A)
GARBA2	B526	D526	D400	C66A	Esegue la GARBAGE COLLECTION-chiude lo spazio nella STRINGA SPAZIO utilizzata per le stringhe e scartate
DVARS	B5BD	D5BD	D497	----	Ricerca le tavole di variabili e matrici per il successivo indicatore di stringa che deve essere salvato dalla GARBAGE COLLECTION
GRBPAS	B606	D606	D40E	----	Riutilizza lo spazio di stringhe cancellate durante la GARBAGE COLLECTION
CAT	B63D	D63D	D517	C74F	Concatena due stringhe in una espressione poi continua la valutazione dell' espressione
MOVINS	B67A	D67A	D554	C78C	Trasferisce la stringa il cui indicatore e' indicato da (STRNG1)
FRESTR	B6A3	D6A3	D57D	C7B5	Conferma del modo stringa poi...
FREFAC	B606	D606	D580	----	Esegue l' inquadramento e l' allocazione della stringa. Inizia con il puntatore all' indicatore della stringa in (FAC+3) ed esce con la lunghezza in (A) e (INDEX1) che punta all' inizio della stringa stessa
FRETMS	B6DB	D6DB	D5B5	C811	Aggiorna l'indicatore di stringa nello SP
CHRD	B6EC	D6EC	D5C6	C822	Punto di ingresso per la funzione CHR\$
LEFTD	B700	D700	D5DA	C836	Punto di ingresso per la funzione LEFT\$
RIGHTD	B72C	D72C	D600	C862	Punto di ingresso per la funzione RIGHT\$
MIDD	B737	D737	D611	C86D	Punto di ingresso per la funzione MID\$

LABEL	C64	V20	B2.0	B4.0	DESCRIZIONE
PREAM	B761	D761	D63B	C897	Estrae dallo Stack il puntatore all' indicator e stringa, lo immagazzina in (DSCPNT), immette il parametro stringa in (a)
LEN	B77C	D77C	D656	C8B2	Punto di ingresso per la funzione LEN
LEN1	B782	D782	D65C	C8B8	Esegue il posizionamento della stringa, dopo forza l' entrata del modo numerico e quindi esce con la lunghezza della stringa in (Y)
ASC	B78B	D78B	D665	C8C1	Punto di ingresso per la funzione ASC, prende il primo carattere nella stringa e lo converte in formato FLTP
GTBYTC	B79B	D79B	D657	C8D1	Valuta un' espressione nel testo. Controllo di validita' per intervallo da 0 a 255, in caso contrario visualizza un "ILLEGAL QUANTITY ERROR". Riporta il valore in (X)
VAL	B7AD	D7AD	D687	C8E3	Punto di ingresso per la funzione VAL. Conferma che l' argomento sia una stringa poi...
STRVAL	B7B5	D7B5	D68F	C8F5	Converte una stringa di inizio a (INDEX1) e di lunghezza (A) in valore FLTP in (FAC)
GETNUM	B7EB	D7EB	D6C6	C921	Legge i parametri dal testo Basic relativamente ai comandi POKE o WAIT, immagazzina il primo intero in (INDEX1), il secondo intero in (INDEX2)
GETADR	B7F7	D7F7	D6D2	C92D	Converte FAC in intero nel (INDEX1) nell' intervallo da 0 a 65535
PEEK	B80D	D80D	D6EB	C943	Punto di ingresso per PEEK
POKE	B824	D824	D707	C95A	Punto di ingresso per POKE
WAIT	B82D	D82D	D710	C963	Punto di ingresso per WAIT
FADDH	B849	D849	D72C	C97F	Somma 0.5 al FAC
FSUB	B850	D850	D733	C986	Sottrazione in virgola mobile: (FAC)=MFLPT a (A/Y)-(FAC)
FSUBT	B853	D853	D736	C989	Punto di ingresso per sottrazione:(FAC)=(ARG)-(FAC)
FADD5	B862	D862	D76E	C998	Parte della routine di normalizzazione della somma
FAD	B867	D867	D773	C99D	Somma in virgola mobile: (FAC)=MFLPT a (A/Y)+(FAC)
FADDT	B86A	D86A	D776	C9A0	Punto di ingresso per la somma:(FAC)=(ARG)+(FAC)
OVER	B97E	D97E	D88A	CAB4	Stampa il messaggio"OVERFLOW ERROR"
MULSHF	B983	D983	D88F	CAB9	Moltiplicazione di un Byte
FONE	B9BC	D9DC	D8C8	CAF2	Costante 1.0 nel formato MFLPT
	B9C1	D9C1	D8CD	CAF7	Diverse costanti usate per la valutazione di funzioni
LOG	B9EA	D9EA	D8F6	CB20	Esegue la funzione LOG-controlla che l' argomento della funzione sia positivo dopo calcola il valore di LOG in base e
FMULT	BA28	DA28	D934	CB5E	Moltiplica il (FAC) per MFLPT il cui indirizzo

LABEL C64 V20 B2.0 B4.0 DESCRIZIONE

					e' contenuto in (A/Y), risposta in (FAC)
FMULTT	BA30	DA30	D934	CB5E	Esegue la routine di moltiplicazione in virgola mobile. Moltiplica (FAC) per (AFAC) il risultato in (FAC)
MLTPLY	BA59	DA59	D965	CB8F	Moltiplica (FAC) per 1 Byte, il risultato in R ESHO
CONUPK	BA8C	DA8C	D998	CBC2	Carica AFAC con il valore di MFLPT indirizzato da (A/Y)
MULDIV	BAB7	DAB7	D9C3	CBED	Subroutine di moltiplicazione per controllare FAC e AFAC per UNDERFLOW/OVERFLOW
MLDVEX	BAD4	DAD4	D9E0	CC0A	Manipola l' errore di OVERFLOW o di UNDERFLOW
MUL10	BAE2	DAE2	D9EE	CC18	Moltiplica il FAC per 10, il risultato in FAC
TENC	BAF9	DAF9	DA05	CC2F	Costante 10 nel formato MFLPT
DIV10	BAFE	DAFE	DA0A	CC34	Divide (FAC) per 10, la risposta in FAC
FDIVF	BB07	DB07	DA13	CC3D	Divide AFAC per il valore MFLPT indirizzato da (A/Y) e il cui segno e' contenuto in X. La risposta viene messa in FAC
FDIV	BBOF	DBOF	DA1B	CC45	Divide AFAC per MFLPT indirizzato da (A/Y). La risposta va in FAC
FDIVT	BB12	DB12	DA1E	CC48	Esegue la routine di divisione in virgola mobile-(AFAC) e' diviso da FAC, la risposta in FAC. All' ingresso della routine (A)=(FACEXP)
MOVFM	BBA2	DBA2	DAAE	CCD8	Carica FAC con MFLPT indirizzato da (A/Y)
MOV2F	BB07	DBC7	DAD3	CCFD	Immagazzina il contenuto di FAC in TEMPF2.
MOV1F	BBCA	DBCA	DAD6	CDO0	Immagazzina il contenuto di FAC in TEMPF1
MOVXF	BBDO	DBDO	BADC	CDO6	Immagazzina il contenuto di FAC nella locazione e indirizzata da (FORPNT)
MOVMF	BBD4	DBD4	DAE0	CDO6	Immagazzina FAC nella locazione il cui indirizzo e' in (X/Y)
MOVFA	BBFC	DBFC	DB08	CD32	Carica FAC da AFAC
MOVAF	BC0C	DC0C	DB18	CD42	Carica AFAC da FAC
ROUND	BC1B	DC1B	DB27	CD51	Arrotonda il contenuto di FAC nel FAC stesso
SIGN	BC2B	DC2B	DB37	CD61	Trova il segno di FAC, il risultato e' immesso in A-\$01=positivo,\$00=zero,\$FF=negativo
SGN	BC39	DC39	DB45	CD6F	Esegue la funzione SGN
ACTOFC	BC3C	DC3C	DB48	CD72	Immagazzina (A) in FAC
INTOFC	BC44	DC44	DB50	CD7A	Immagazzina un intero presente in (FAC+1) come FLTP nel FAC. In ingresso X dovrebbe contenere \$90
ABS	BC58	DC58	DB64	CD8E	Esegue la funzione ABS
FCOMP	B5CB	DC5B	DB67	CD91	Confronta (FAC) con MFLPT indirizzato da (A/Y) il risultato e' riportato in A : \$0=FAC maggiore del valore di MFLPT, \$00=al valore di MFLPT, \$FF=FAC minore di FLMPT
QINT	BC9B	DC9B	DBA7	CDD1	Converte un valore FLPT presente in FAC in quattro Bytes interi in FAC+1 nella forma HI LOW
INT	BCCC	DCCC	DBD8	CE02	Esegue INT- converte FAC in intero poi lo ricalcola

LABEL C64 V20 B2.0 B4.0 DESCRIZIONE

					nverte in FLPT e lo rimette in FAC
FIN	BCF3	DCF3	DBFF	CE29	Converte una stringa ASCII, il cui indirizzo nel testo BASIC e' indicato da TXTPTR, a FLPT nel FAC
	BDB3	DDB3	DCBF	CEE9	Costante MFLPT usata nella conversione in stringa ASCII
INPRT	BDC2	DDC2	DCCE	CF78	Stampa "IN" seguito dall' attuale numero di linea. Esempio (CURLIN).
LINPRT	BDCD	DDCD	DCD9	CF83	Stampa l'attuale numero di linea da (CURLIN)
FACOUT	BDD7	DDD7	DCE3	CF8D	Stampa FAC come stringa ASCII
FOUT	BDDD	DDDD	DCE9	CF93	Converte FAC in stringa ASCII con inizio allo STACK e fine con Byte nullo
FYOUT	BDDF	DDDF	DCEB	CF95	Converte FAC in stringa ASCII di inizio a STAC K-1+Y
FOUTIM	BE68	DE68	DD74	D01E	Converte TI in stringa ASCII con partenza a ST ACK e fine con un Byte nullo.
	BF11	DF11	DE1D	D067	Costanti MFLPT usate in conversione ASCII
SQR	BF71	DF71	DE5E	D108	Esegue la funzione SQR
FPWRT	BF7B	DF7B	DE68	D112	Esegue l' Esponenziale. AFAC alla potenza di F AC risposta in FAC
NEGOP	BFB4	DFB4	DEA1	D14B	Negazione di FAC, risposta in FAC
	BFBF	DFDF	DEAC	D156	Costanti MFLPT per la routine EXP
EXP	BFED	DFED	DEDA	D184	Valuta la funzione EXP
POLYX	EO43	EO40	DF2D	D1D7	Valutazione ddi funzione. All' ingresso (A/Y) punta al singolo Byte che e' uguale a 1- il numero di costanti che lo segue. La routine inizialmente converte l' argomento in un intervallo compreso fra 0 e 0.999999999.
	EO8D	EO8A	DF77	D221	Costanti MFLPT per valutazione RND
RND	EO97	EO94	DF7F	D229	Esegue RND
BOERR	EOF9	EOF6	----	----	Manipola un errore di I/O entro il basic
BCHOUT	E10C	E109	----	----	Routine di uscita carattere basic - usa la routine Kernal CHROUT
BCHIN	E112	E10F	----	----	Routine di ingresso carattere basic - usa la routine Kernal CHRIN
BCKOUT	E118	E115	----	----	Routine di apertura di un canale basic in uscita - usa la routine Kernal CKOUT
BCKIN	E11E	E11B	----	----	Routine basic di apertura canale per INPUT - usa la routine Kernal CHKIN
BGETIN	E124	E121	----	----	Routine di lettura di un carattere basic - usa la routine Kernal GETIN
SYS	E12A	E127	FFDE	FFDE	Esegue SYS - CBM64/VIC20 fissano (A,X,Y e SR) rispettivamente da (SYSX),(SYSY),(SYSS) prima di entrare nella routine in codice macchina e ripristinano i nuovi valori al ritorno
SAVET	E156	E153	FFDB	FFDB	Esegue SAVE - CBM64 e VIC20 ricercano i parametri dal testo basic prima della chiamata della

					routine Kernal, anche la routine Kernal del Pe t legge i parametri
SAVER	E15F	E15C	F6A7	F6E3	Esegue il salvataggio della Ram su una data pe riferica - il CBM64 e il VIC20 saltano alla routine Kernal SAVE
VERFYT	E165	E162	FFDB	FFDB	Esegue VERIFY - il CBM64 e il VIC20 ricercano i parametri dal testo basic prima di saltare a lla routine Kernal.
LOADT	E168	E165	FFD5	FFD5	Esegue LOAD - il CBM64 e il VIC20 ricercano i parametri dal testo basic prima di saltare a lla routine Kernal.
LOADR	E175	E172	F322	F356	Carica la RAM da una periferica - il CBM64 e i l VIC20 ricercano i parametri dal testo basic prima di saltare alla routine Kernal.
OPENT	E1BE	E1BB	FFC0	FFC0	Esegue OPEN - il CBM64 e il VIC20 ricercano i parametri dal testo basic prima di saltare a lla routine Kernal.
CLOSET	E1C7	E1C4	FFC3	FFC3	Esegue CLOSE - il CBM64 e il VIC20 ricercano i parametri dal testo basic prima di saltare a lla routine Kernal.
CLOSER	E1CA	E1C7	F2AC	F2E0	Chiude un dato FILE
SLPARA	E1D4	E1D1	F43E	F47D	Prende i parametri dal testo basic per LOAD/SA VE/VERIFY
COMBYT	E200	E1FD	F460	F49F	Se (TXTPTR) e' indirizzato a una apice , leg ge un byte dal testo basic
DEFLT	E206	E203	F50E	F54D	Se viene trovato la fine di un comando esce se nza aver fissato i parametri.
CMMERR	E20E	E20B	F516	F555	Verifica che TXTPTR indirizzato all' apice n on sia seguito da virgola o da byte nullo - in caso contrario SYNTAX ERROR
OCPARA	E219	E216	F4CE	F50D	Ricerca i prametri dal testo Basic per le rout ines di OPEN e CLOSE
COS	E264	E261	DFD8	D282	Valuta la funzione COS- aggiunge pi greco/2 a FAC poi...
SIN	E26B	E268	DFDF	D289	Valuta la funzione SIN
TAN	E2B4	E2B1	E028	D2D2	Valuta la funzione TAN calcolando SIN/COS
PI2	E2E0	E2DD	E054	D2FE	Costante pi greco/2 di MFLPT
TWOPI	E2E5	E2E2	E059	D303	Costante 2*pi greco di MFLPT
FR4	E2EA	E2E7	E05E	D308	Costante 0.25 di MFLPT
	E2EF	E2EC	E063	D30D	Costante MFLPT per il calcolo della funzione S IN
ATN	E30E	E30B	E08C	D32C	Calcola la funzione ATN
	E33E	E33B	E0BC	D35C	Costante MFLPT per il calcolo di ATN
BASSFT	E37B	E467	----	----	Routine di restart (WARM) del Basic chiamata c on un BREAK se una istruzione BRK e' rilevata oppure se e' premuto RUN/STOP RESTORE. Chiude i canali, resetta le linee di I/O, resetta lo

LABEL C64 V20 B2.0 B4.0 DESCRIZIONE

					STACK ed esce attraverso la routine IERROR con (X) = \$80
INITV	E453	E45B	----	----	Copia BVTRS nel blocco 0 della RAM
INIT	E394	E378	----	----	Inizializza il Basic all' accensione-se viene chiamata INITV per settare i vettori Basic in \$0300-\$030B, allora....
ININV	E397	E37B	E116	D3B6	Chiama la routine di INITCZ per settare le variabili Basic nella pagina zero. (Il PET azzerà il contenuto della memoria da \$0400 a fine memoria, mentre cio' non'avviene con CBM64 e VIC20). Chiama in funzione la routine INTMS, poi v a in modo Basic.
INITAT	E3A2	E387	E0F9	D399	Copia principale della routine CHRGET - copia all' indietro in pagina ZERO da INITCZ
RNDSER	E3BA	E39F	E111	D3B1	Costante MFLPT=0.811635157 usata per il valore iniziale della funzione RND
INITCZ	E3BF	E3A4	----	----	Inizializza la RAM Basic- fissa USRPOK, ADRA1, ADRA2, copia INITAT e RNDSER in CHRGET e RNDX, fissa TXTTAB e FRETOP a (LORAM), (HIRAM), mette a zero il primo Byte del testo Basic
WORDS	E460	E429	E1B7	D44B	Testo "BYTES FREE"
FREMES	E473	E436	E1C4	D458	Testo"...COMMODORE BASIC..."
BVTRS	E447	E44F	----	----	Copia ROM dei vettori BASIC
IOBASE	E500	E500	----	----	Riporta in (X/Y) l' indirizzo del 6526 (CIA) utilizzato dalla routine IRQ. Questa e' la routine Kernel IOBASE
SCRNK	E505	E505	----	----	Riporta la disposizione organizzativa dello schermo-colonne (X), righe (Y). Entrata attraverso il vettore Kernel SCREEN
PLOTK	E50A	E50A	----	----	Fissa/riporta la posizione del cursore: le colonne di schermo attraverso (Y), le righe di schermo attraverso (X). Fissa il cursore se il Carry e' a 0, ne riporta invece la posizione se il Carry e' a 1. Ingresso attraverso il vettore Kernel PLOT.
INITIO	E518	E518	E1DE	E60F	Inizializza I/O. Questa routine e' chiamata tramite il vettore Kernel IOBASE
HOME	E566	E581	E257	E05F	Home di schermo, resetta la tavola di link delle linee di schermo
PLOTR	E56C	E587	E25D	E06F	Muove il cursore a (TBLX), (PNTR)
PANIC	E59A	E5B5	----	----	Resetta le linee di I/O, incluso i registri del VIC-II
DLTIO	E5A0	E5BB	----	----	Resetta i canali di I/O poi....
VICINT	E5A8	E5C3	----	----	Ripristina i valori del 6567 (VIC-II)
KBGET	E5B4	E5CF	E285	E0A7	Prende un carattere dal Buffer di tastiera
KBINP	E5CA	E5E5	E29A	E0BC	Input di un carattere
KSINP	E632	E64F	E2F4	E116	Input di un carattere da tastiera o schermo

LABEL C64 V20 B2.0 B4.0 DESCRIZIONE

SCINP	E63A	E657	E2FF	E121	Input di un carattere dallo schermo.
TGLQT	E684	E688	E33F	E16A	Congiunge il flag di virgolette (QTSw). Durant e la fase di INPUT arresta la codificazione (T OKEN) delle parole chiave quando queste sono f ra apici
SCPUT	E691	E6C5	E34C	E177	Stampa il contenuto di (A)sullo schermo. Utili zzata da SCNPNT
SCNPNT	E716	E742	E3D8	E202	Stampa un carattere sullo schermo, interpretan do sia i controlli di cursore, il cambio di co lore e il cambio di set di caratteri
CKDECL	E8A1	E8E8	----	----	Controlla il decremento del contatore linea.
CKINL	E8B3	E8FA	----	----	Controlla l' incremento del contatore linea.
SCNTABL	E8DA	E921	----	----	Tavola usata per la decodifica dello schermo
SCROLL	E8EA	E975	E53F	E3C8	Routine di scrolling schermo
IRQK	EA31	EA41	E61B	E442	Routine principale di manipolazione interrupt di IRQ.(CINV punta a queste locazioni di memo ria)
SCNKIK	EA87	EB1E	E68E	E4CD	Routine di scansione della tastiera, controllo di tasto premuto e immette il carattere nella coda di tastiera. Questa e' la routine indiriz zata dal vettore Kernal SCNKEY
KBDTBL	EB79	EC46	E6F8	E6D1	Tavola della matrice di tastiera. Utilizzata d a SCNKYK per convertire il tasto premuto nel c arattere ASCII
TALKK	ED09	EE14	FOB6	F0D2	Esegue un OR con (A) per convertire il numero di periferica in un indirizzo di TALK per il B US IEEE e lo trasmette come comando. Questa e' la routine Kernal indirizzata da TALK
LSTNK	EDOC	EE17	FOBA	F0D5	Esegue un OR con (A) per convertire il numero di periferica in un indirizzo di LISTEN per il bus IEEE e lo trasmette come comando.Questa e' la routine Kernal indirizzata da LISTEN
SCNDK	EDB9	EECO	----	----	Converte (A) e lo trasmette come indirizzo sec ondario di LISTEN sul BUS IEEE. Questa e' la routine Kernal a cui si accede per mezzo di SE COND
TKSAK	EDC7	EECE	----	----	Converte (A) e lo trasmette come indirizzo sec ondario di TALK sul BUS IEEE. Questa e' la routine Kernal a cui si accede per mezzo di TK SA
CIOUTK	EDDD	EEE4	F16F	F19E	Trasmette un Byte sul BUS IEEE. Il carattere e ' bufferizzato cosi' che e' possibile eseguire un HANDSHAKING.Questa e' la CIOUT Kernal
UNTLKK	EDEF	EEF6	F17F	F1AE	Trasmette un comando di UNTALK sul BUS IEEE. Q uesta e' la routine Kernal che e' indirizzata dal vettore UNTALK
UNLSNK	EDFE	EFO4	F183	F1B9	Trasmette un comando UNLISTEN sul BUS IEEE. Il

LABEL C64 V20 B2.0 B4.0 DESCRIZIONE

ACPTRK	EE13	EF19	F18C	F1C0	vettore Kernal UNLSN comincia di qui Un byte e' in HANDSHAKEN dal BUS IEEE e immesso in A. Questa e' la routine Kernal ACPTR
NMICNT	EEBB	EFA3	----	----	Continuazione della Routine principale di INTERRUPT NMI usata per le periferiche RS232
RSWRT	EF06	EFEE	----	----	Uscita di un Byte su canale RS232
RSBLD	EF59	F036	----	----	Parte della Routine di interrupt NMI che costruisce il bit individuale proveniente dal canale RS232
KMSGTX	F0BD	F174	F000	F000	Testo relativo ad un errore Kernal e controllo del messaggio ivi immagazzinato.
KMESSG	F12B	F1E2	F156	F158	Stampa un messaggio Kernal sullo schermo
GETINK	F13E	F1F5	F1D1	F205	Riceve un carattere da un canale e lo riporta in A. Se nessun carattere e' stato inviato riporta 0. Questa e' la routine Kernal GETIN
CHRIK	F157	F20E	F1E1	F215	Input di un carattere dal Buffer in A. Questa e' la routine Kernal CHRIN
CHROTK	F1CA	F27A	F232	F266	Uscita del Byte contenuto in A nel canale di uscita. Questa e' la routine Kernal CHROUT
CHKINK	F20E	F2C7	F7BC	F7FE	Colloca il file dichiarato da X come canale di input. Questa e' la routine usata dal vettore Kernal CHKIN
CKOUTK	F250	F309	F770	F71F	Colloca il file dichiarato da Y come canale di uscita. Questa e' la routine usata dal vettore Kernal CHKOUT
CLOSEK	F291	F34A	F2AC	F2E0	A dichiara il file che deve essere chiuso. I particolari sono rimossi dalla tavola delle periferiche (SAT, FAT). Questa e' la routine Kernal CLOSE
CLALLK	F32F	F3EF	F26E	F2A2	Questa routine blocca tutti gli attuali I/O. Il numero di file aperto (LNTND) e' messo a zero e qualsiasi file su IEEE e' messo in UNTALK o UNLISTEN. Questa e' la routine Kernal CLALL.
CLRCHK	F333	F3F3	F272	F2A6	Dealloca i canali di I/O e rifissa le periferiche (DFLNT=0,DFLTO=3). Questa e' la routine Kernal CLRCHN
OPENK	F34A	F40A	F524	F563	Apre il file i cui dati sono immagazzinati in FNLEN,LA,SA,FNADR inserendo i dettagli nelle tavole di LAT,FAT e SAT e scegliendo le appropriate routine per la gestione di files su disco o cassetta. Questa e' la routine Kernal OPEN
LOADK	F49E	F542	F3CC	F40B	Carica il file negli indirizzi detti sopra. Il CBM64 ed il VIC 20 hanno un parametro addizionale che dichiara dove il file deve essere ricaricato. Questa e' la routine Kernal LOAD
SAVEK	F5DD	F675	F6A4	F6E3	Salva una data zona di memoria RAM (STAL e NEM

LABEL C64 V20 B2.0 B4.0 DESCRIZIONE

UDTIMK	F69B	F734	F729	F768	USS) su un file indetificato da (FNLEN,LA,FA, SA,FNADR). Questa e' la routine kernal SAVE Parte della routine di servizio IRQ che posiziona l' orologio in tempo reale. Inoltre immagazzina l'attuale valore della matrice di tastiera in STKEY che abilita la funzione di STOP. Questa e' la routine Kernal UDTIM
STOPK	F6ED	F770	F30F	F343	Controlla il valore immagazzinato in STKEY e restituisce il flag Z a 1 se il valore ivi immagazzinato rappresenta il tasto STOP. Questa e' la routine Kernal STOP
KERROR	F6EB	F77E	F315	F349	Errori controllati dalle routines Kernal. Questa routine entra in funzione per inviare il giusto messaggio di errore
THEADR	F72C	F7AF	F5A6	F5E5	Cerca e legge il bloccodi testa del nastro.
TCNTL	F80D	F88A	F806	F84B	Le routines di controllo nastro risiedono in questo punto. Eseguono le funzioni come: controllo del motore di cassetta, temporizzazione, ecc.
TREAD	F92C	F98A	F855	F89A	Routine di lettura nastro
TBYT	FA70	FABD	F931	F976	Routine di manipolazione di un Byte per lettura nastro
TWRT	FBA5	FBEA	F893	FBD8	Routine di scrittura nastro
COLD	F6E2	FD22	FCD1	FD16	Routine di inizializzazione utilizzata all'accensione della macchina. Questa e' la routine che e' indirizzata via HARDWARE dal vettore localizzato a \$FFFC. La memoria e' inizializzata e tutte le periferiche di I/O sono fissate. La prima parte della routine controlla se un cart ridge e' caricato nel blocco 9 e se e' cosi' salta al cartridge per la inizializzazione e la messa in funzione
MEMCHK	----	FE91	----	----	Questa routine nel VIC20 controlla quanta memoria e' disponibile
NMIXCT	FE43	FEA9	----	----	Nel VIC e nel CBM 64 l' NMI Interrupt e' principalmente usato per gestire le periferiche RS232. I PET invece non l' adoperano per niente
BAUDD	----	FF5C	----	----	Tavola dei BAUD RATE per il VIC
INTRPT	FF48	FF72	E61B	E442	Questa routine entra in funzione quando viene rilevato un Interrupt. I registri sono salvati e la causa dell' Interrupt viene determinata (IRQ o BRK)
KVCTRS	FFB1	FF8A	FFC0	FFC0	Tavola dei salti delle routines Kernal
NMIVEC	FFFA	FFFA	FFFA	FFFA	Vettore di Interrupt NMI
RSTVEC	FFFC	FFFC	FFFC	FFFC	Vettore di Reset
IRQVEC	FFFE	FFFE	FFFE	FFFE	Vettore di Interrupt IRQ

NOME

VIA

CITTA'

SONO IN POSSESSO DI

CON LE SEGUENTI PERIFERICHE

☐ Desidero ricevere gratuitamente il Vostro
catalogo ed essere inserito nell' E.V.M.
MAILING LIST

☐ Desidero informazioni sulla SOFTWARE BANK

SEGNALAZIONE DI ERRORI

SPEDIRE IN BUSTA A:

E.V.M Computers
Via Marconi 9/a
52025 MONTEVARCHI (AR)
